TR-I-0270

# HMM-LR ソース・コード

北　研二

ATR Interpreting Telephony Research Laboratories

1992.8

## 概要

本資料は、ATR で開発した HMM-LR 音声認識システムのソース・コードである。HMM-LR は、HMM(Hidden Markov Model) と LR パーザを統合化した音声認識システムであり、言語の構文情報を用いて効率よく入力された音声を認識することができるという特徴を持つ。本資料では、著者が作成を担当した LR の部分に関しては、日本語のコメントが付けられており、プログラムの理解を極めて容易にしている。

ATR 自動翻訳電話研究所
ATR Interpreting Telephony Research Laboratories

# HMM-LRソース・コード

北　研二
ATR自動翻訳電話研究所

## 1　はじめに

本資料は、ATRで開発したHMM-LR音声認識システムのソース・コードである。HMM-LRは、HMM (Hidden Markov Model) とLRパーザを統合化した音声認識システムであり、言語の構文情報を用いて効率よく入力された音声を認識することができるという特徴を持つ。

本資料では、著者（北　研二）が作成を担当したLRの部分に関しては、日本語のコメントが付けられており、プログラムの理解を極めて容易にしている。

## 2　ソース・コードについて

ソース・コードは、マシン atr-dp 上の下記のディレクトリにある。

/ifg2/LR/HMM-LR/Hmmlr.V7J

実行ファイルは、同じディレクトリ上にある Makefile を実行することにより、作ることができる。

## 3 各ファイルの概要

| | | |
|---|---|---|
| Makefile | ··· | ソース・ファイルから実行ファイルを作成する。 |
| config.h | ··· | システムの configuration の定義。 |
| hmmlr.h | ··· | 各種大域変数の定義。 |
| cell.h | ··· | 認識仮説を格納するセルの構造体定義。 |
| main.c | ··· | HMM - ＬＲのメイン・プログラム。<br>オプション解析、初期化等を行なう。 |
| hmmlr.c | ··· | HMM - ＬＲの音声認識部。 |
| cell.c | ··· | 認識仮説を格納するセルに対しての操作。 |
| verify.c | ··· | HMM音韻照合の駆動。 |
| LR.h | ··· | 文法、ＬＲ表を格納する構造体等の定義。 |
| LR.c | ··· | 文法、ＬＲ表のロード。 |
| ReadLR.c | ··· | ファイルからＬＲ表を読み込む。 |
| ReadRules.c | ··· | ファイルから文法を読み込む。 |
| fileio.c | ··· | ファイルからの入力。 |
| beam.c | ··· | ビームサーチを実行する。 |
| time.c | ··· | ＣＰＵ時間、経過時間の測定用関数。 |
| error.c | ··· | エラー・メッセージの表示。 |
| LRctrl.c | ··· | 文節間ＬＲパーザ。 |
| ReadLR2.c | ··· | ファイルから文節内ＬＲ表を読み込む。 |
| int_op.c | ··· | 文節カテゴリのロード。 |
| idur.h | ··· | 継続時間制御用の定義。 |
| imhmm.h | ··· | HMM音韻モデルの定義。 |
| fuzzy.h | ··· | ファジーＶＱ用の定義。 |
| log.h | ··· | log compression table の定義。 |
| adapt.c | ··· | 話者適応を行なう。 |
| calidm.c | ··· | HMMの確率計算を行なう。 |
| idur.c | ··· | 継続時間長の制御を行なう。 |
| init_adapt.c | ··· | HMM音韻モデル、継続時間長パラメータのロード。 |
| io_mhmm.c | ··· | ファイルからHMM音韻モデルの読み込む。 |
| dtoi.c | ··· | データのＤＥＣ→ＩＥＥＥ変換を行なう。 |
| logtbl.c | ··· | log compression table の読み込み。 |
| hash.h | ··· | ハッシュ・テーブルの定義。 |
| hash.c | ··· | 音節 trigram 使用時のハッシュ関数。 |
| ngram.h | ··· | 音節 trigram データを格納するテーブルの定義。 |
| calcprob.c | ··· | 統計的言語モデルを用いた仮説の尤度計算。 |
| StochasticGrammar.c | ··· | 確率文法の初期化。 |
| ngram.c | ··· | 音節 trigram の読み込み、確率計算。 |

```
#
# Makefile for HMM-LR Speech Recognizer
#
CFLAGS = -O

LIBS = -lm -ltermcap

INCLUDES = config.h cell.h HMM.h LR.h imhmm.h idur.h log.h fuzzy.h

TARGET  = Hmmlr

OBJECTS =   main.o hmmlr.o ReadRules.o ReadLR.o LR.o cell.o init_adapt.o error.o\
            time.o io_mhmm.o idur.o logtbl.o calidm.o adapt.o\
            beam.o verify.o fileio.o calcprob.o ngram.o hash.o\
            LRctrl.o ReadLR2.o int_op.o StochasticGrammar.o dtoi.o

SWITCHES = -o $(TARGET) $(LIBS)

$(TARGET): $(OBJECTS)
        cc $(CFLAGS) $(OBJECTS) $(SWITCHES)

$(OBJECTS): $(INCLUDES)
```

LINE # HEADER TEXT

```
/*----------------------------------------------------------------------
 *
 *      HMM関連の変数等の定義
 *
 *          ATR Interpreting Telephony Research Laboratories.
 *
 *      History
 *
 *          7-Sep-1988  Created by Toshiyuki Hanazawa and Kenji Kita.
 *
 *----------------------------------------------------------------------*/

#include "imhmm.h"
#include "idur.h"
#include "fuzzy.h"

/*
 * 入力音声の最大フレーム長．
 */
#define MAXFRAME            1024

/*
 * 1フレームの長さ（デフォルト値）．
 */
#define  FRAME_PERIOD        3  /* msec */

/*
 * 音声データを読み込む際に，前後に付け加える無音区間の長さ
 * （デフォルト値）．
 */
#define  ADD_TIME           54  /* msec */

/*
 * 変数 end_free のデフォルト値．
 * 最終的な認識候補の尤度は，音声の終端から end_free フレーム
 * 以内にある各フレームの正規化尤度のうち，最もよいものが採用される．
 */
#define  END_FREE           27  /* msec */

/*
 * 継続時間長パラメータへの数の上限．
 */
#define  MAX_NUM_DUR         2

/*
 * HMM 音韻モデル，継続時間長パラメータに関する情報．
 */
typedef struct Entry
{
        char    *phone;         /* 音韻モデル名 */
        IMHMM   *model;         /* 音韻モデル，パラメータへのポインタ */
        IDUR    *duration[MAX_NUM_DUR];
                                /* 継続時間長パラメータへのポインタ */
        int     num_dur;        /* 継続時間長パラメータの数 */
        struct Entry    *next;  /* 次の Entry へ */
}       Entry;

/*
 * 認識対象となる音声データの総フレーム数．
 */
#ifndef HMM_MAIN
extern
#endif
int     N_frames;

/*
 * 以下，枝沢コード．
 */
#ifndef HMM_MAIN
extern
#endif
FDATA   Fdata[MAX_CODEBOOK];

#ifndef HMM_MAIN
extern
#endif
int     leng[MAX_CODEBOOK], Mabiki;

#ifndef HMM_MAIN
extern
#endif
double  Sigma_model, Sigma_state, Nsd, Dur_weight;

#ifndef HMM_MAIN
extern
#endif
double  Frame_Period, Add_Time, End_Free;

#ifndef HMM_MAIN
extern
#endif
int     Mabiki;
```

LINE # HEADER TEXT

```
/*----------------------------------------------------------------------
*
*       LRパーザ用の定義
*
*       ATR Interpreting Telephony Research Laboratories.
*
*       History
*
*           23-Jun-1988. Created by Kenji Kita.
*           23-Aug-1988. Modified.
*           29-Jun-1991. Two-level version
*
----------------------------------------------------------------------*/

/*
 * シンボル数の上限．
 * MAXSYMBOLS は素数でなければならない．
 */
#define MAXSYMBOLS      1999

/*
 * 文法数の上限．
 */
#define MAXGRAMMARS     10000

/*
 * LR解析表の状態数の上限．
 */
#define MAXSTATES       15000

/*
 * LR解析表の中での入力の終りを表す記号．
 */
#define ENDSYMBOL       "$"

/*
 * LR解析表（Action 構造体）中で，受理動作を示す．
 */
#define ACCEPT          (-2)

/*
 * 還元，移動，GOTO，受理動作か？
 */
#define IsREDUCE(n)     (n >= 2*MAXSTATES ? 1 : 0)
#define IsSHIFT(n)      (n >= 0 && n < MAXSTATES ? 1 : 0)
#define IsGOTO(n)       (n >= MAXSTATES && n < 2*MAXSTATES ? 1 : 0)
#define IsACCEPT(n)     (n == ACCEPT ? 1 : 0)

/*
 * 還元動作の場合の文法規則の番号．
 */
#define REDUCE(n)       ((n)-2*MAXSTATES)

/*
 * 移動動作の場合の新しい状態番号．
 */
#define SHIFT(n)        (n)

/*
 * GOTO の行き先．
 */
#define GOTO(n)         ((n)-MAXSTATES)

/*
 * 文法規則を記述する構造体．
 */
typedef struct Rule
{
#if STOCHASTIC_GRAMMAR
        double  gprob;          /* 規則の適用確率 */
#endif
        int     length;         /* 右辺の長さ（シンボル数） */
        int     lhs;            /* 左辺 */
        int     *rhs;           /* 右辺 */
}       Rule;

/*
 * LR解析表を記述する構造体．
 */
typedef struct Action
{
        int     input;          /* 入力シンボル */
        int     action;         /* 動作 */
        struct Action   *next;  /* 同じ状態での，次の動作項へのポインタ */
#if STOCHASTIC_LR
        double  prob;           /* 確率 */
#endif
#if TWO_LEVEL
        int     *cat;           /* 到達可能な文法カテゴリ */
#endif
}       Action;

/*
 * シンボル・テーブル．
 * プログラム中では，すべてのシンボル（音韻名，文法の終端，非終端記号等）は，
 * 負でない整数で表されている．
 * 整数 n に対応する記号は，SymbolTable[n] に格納される．
 */
#ifndef LR_MAIN
extern
#endif
char    *SymbolTable[MAXSYMBOLS];
```

LINE # HEADER TEXT

```
/*----------------------------------------------------------------------*
 *
 *      セルの定義
 *
 *          ATR Interpreting Telephony Research Laboratories.
 *
 *      History
 *
 *          24-Jun-1988, Created by Kenji Kita.
 *          29-Jun-1991, Two-level version.
 *
 *----------------------------------------------------------------------*/

/*
 * 総セル数（デフォルト値）。
 */
#define TOTALCELLS          512

/*
 * ビーム幅（デフォルト値）。
 */
#define GLOBAL_BEAM         100

/*
 * 局所的なビーム幅（１つの仮説からの分岐数）（デフォルト値）。
 */
#define LOCAL_BEAM          15

/*
 * LRパーザの状態スタックの大きさ。
 */
#define LR_STACKSIZE        100

/*
 * 認識された音韻列格納用スタックの大きさ。
 */
#define IN_STACKSIZE        100

/*
 * 文法履歴スタックの大きさ。
 */
#define G_STACKSIZE         100

/*
 * 単語境界を示すデリミタ。
 */
#define WORD_DELIMITER      (-1)

/*
 * セル cp のLR状態スタックの最上段の状態を調べる。
 */
#define stacktop(cp)    cp->stack[cp->stackptr-1]

/*
 * セル cp のLR状態スタックに n をプッシュする。
 */
#define push(n, cp)     cp->stack[(cp->stackptr)++] = n

/*
 * セル cp のLR状態スタックからポップする。
 */
#define pop(cp)         cp->stack[--(cp->stackptr)]

/*
 * セル cp のLR状態スタックから n 個の状態をポップする。
 */
#define npop(n, cp)     cp->stack[(cp->stackptr -= n)]

/*
 * セル cp のLR状態スタックをクリア。
 */
#define clrstack(cp)    cp->stackptr = 0

/*
 * 以下の４行は、音韻列格納用スタックに対する操作。
 */
#define instacktop(cp)  cp->instack[cp->instackptr-1]
#define inpush(n, cp)   cp->instack[(cp->instackptr)++] = n
#define inpop(cp)       cp->instack[--(cp->instackptr)]
#define clrinstack(cp)  cp->instackptr = 0

/*
 * 以下の２行は、文法履歴格納用スタックに対する操作。
 */
#define gpush(n, cp)    cp->gstack[(cp->gstackptr)++] = n
#define clrgstack(cp)   cp->gstackptr = 0

/*
 * 以下の２行は、音節 trigram 用スタックに対する操作。
 */
#define mpush(n, cp)    cp->mstack[(cp->mstackptr)++] = n
#define clrmstack(cp)   cp->mstackptr = 0

/*
 * 未使用。
 */
#define cellcopy(p, q)  bcopy((char *)p, (char *)q, sizeof(Cell))

/*
 * セルの構造体の定義。
 * セルとは、認識された仮説に関する情報を記憶しておくための構造体である。
 */
typedef struct Cell
{
        /*
         * LRの状態スタックおよびスタック・ポインタ。
         */
        int     stack[LR_STACKSIZE];
        char    stackptr;

        /*
         * 認識された音韻列格納用スタック。
         */
        int     instack[IN_STACKSIZE];
        char    instackptr;

#if GTRACE
        /*
         * 文法履歴格納用スタック。
         */
```

LINE # | HEADER TEXT

```
121         int     gstack[G_STACKSIZE];
122         char    gstackptr;
123 #endif
124
125 #if SYLLABLE_NGRAM
126         /*
127          * 音節 trigram を用いて、尤度を計算するための変数定義.
128          */
129         int     mstack[IN_STACKSIZE];   /* Mora index */
130         int     mstackptr;
131         char    symbol[6];
132         double  moraprob;
133 #endif
134
135 #if L_MODEL
136         /*
137          * HMMだけを用いたときの、仮説の尤度.
138          * 統計的言語モデルを用いているときだけ使用.
139          */
140         double  hmmprob;
141 #endif
142
143 #if BEST_FIRST
144         /*
145          * Best-first 探索では、最良の仮説に対して音韻照合が起こるが、
146          * この音韻照合が行なわれたセルに対して、変数 shifted がセットされる.
147          */
148         char    shifted;
149 #endif
150
151 #if TWO_LEVEL
152         /*
153          * 2段階LRの場合、認識された仮説に対する文節カテゴリ.
154          */
155         int     topcat;
156 #endif
157
158         /*
159          * 以下の部分は、HMMの音韻照合で用いる.
160          * start     ... 音声照合区間の始端.
161          * end       ... 音声照合区間の終端.
162          * bestprob  ... 最大正規化確率.
163          * bestpoint ... 最大正規化確率を持つフレーム番号.
164          * prob[]    ... 尤度格納用の配列.
165          */
166         int     start;
167         int     end;
168         int     bestprob;
169         int     bestpoint;
170         int     prob[MAXFRAME+1];
171
172         /*
173          * 以下の部分は、上述のものと同様.
174          * ただし、音声の最後の母音の照合のときに用いる.
175          */
176         char    _flag;
177         int     _start;
178         int     _end;
179         int     _bestprob;
180         int     _bestpoint;
181         int     _prob[MAXFRAME+1];
182
183         /*
184          * 次にセルへのポインタ.
185          */
186         struct Cell     *next;
187 }       Cell;
188
189 #ifndef CELL_MAIN
190 extern
191 #endif
192 int     globalbeam,     /* ビーム幅 */
193         localbeam,      /* 局所的なビーム幅（1つの仮説からの最大分岐数）*/
194         totalcells;     /* 確保するセルの総数 */
195
196 #ifndef CELL_MAIN
197 extern
198 #endif
199 Cell    *cellstart,     /* セル領域の先頭 */
200         *cellend,       /* セル領域の最後 */
201         *celltop,       /* 認識仮説リスト */
202         *newcelltop,    /* 更新された認識仮説リスト */
203         *freecellp,     /* 自由セルのリスト */
204         *acceptedcellp, /* 受理されたセルのリスト */
205         *localcellp;    /* 1つのセルから分岐したセルのリスト */
```

LINE # HEADER TEXT

```
1  /*-----------------------------------------------------------------------*
2   *                                                                       *
3   *      System Configuration                                             *
4   *                                                                       *
5   *        ATR Interpreting Telephony Research Laboratories.              *
6   *                                                                       *
7   *        Created by Kenji Kita and Toshiyuki Hanazawa.                  *
8   *                                                                       *
9   *-----------------------------------------------------------------------*/
10
11 /*
12  * プログラムで使用する binary データを読み込む際に，
13  * DEC -> IEEE 変換を行なうか？
14  */
15 #define DEC_TO_IEEE             1
16
17 /*
18  * HMMの状態ごとの継続時間長制御を行なうか？
19  */
20 #define DURATION_CONTROL        1
21
22 /*
23  * HMMの確率計算にTrellisあるいはViterbiのどちらを用いるか？
24  */
25 #define TRELLIS                 1
26 #define VITERBI                 !(TRELLIS)
27
28 /*
29  * 認識時に適用された文法規則の履歴を保持するか？
30  * 確率文法を使用するときには，GTRACEを１にすること．
31  */
32 #define GTRACE                  1
33
34 /*
35  * 音節の３つ組確率を認識時に用いるか？
36  */
37 #define SYLLABLE_NGRAM          1
38
39 /*
40  * 確率文法を認識時に用いるか？
41  */
42 #define STOCHASTIC_GRAMMAR      1
43
44 /*
45  * 確率付きLR解析表を認識時に用いるか？
46  */
47 #define STOCHASTIC_LR           0
48
49 /*
50  * Best-first探索を行なうか？
51  */
52 #define BEST_FIRST              0
53
54 /*
55  * ２段階LRを用いるか？
56  */
57 #define TWO_LEVEL               0
58
59 /*
60  * 統計的言語モデル使用時は，L_MODEL が１にセットされる．
61  */
62 #if SYLLABLE_NGRAM | STOCHASTIC_GRAMMAR | STOCHASTIC_LR
63 #  define L_MODEL               1
64 #else
65 #  define L_MODEL               0
66 #endif
```

LINE # | HEADER TEXT

```
1  #define     MAX_FUZZY           6
2
3  struct fuzzy_data
4  {
5    int               num_fuzzy, length;
6    float             fuzziness;
7    unsigned char     *codo;
8    float             *m;
9  };
10
11 typedef  struct fuzzy_data  FDATA;
```

LINE # HEADER TEXT

```
/*-----------------------------------------------------------------------
 *
 *      ハッシュ・テーブルの定義
 *
 *      このファイルでは、一般的なハッシュ・テーブルの定義が
 *      与えられているが、音節 trigram のプログラムでしか
 *      使用されていない。
 *
 *          ATR Interpreting Telephony Research Laboratories.
 *
 *      History
 *          4-Jul-1990. Created by Kenji Kita
 *
 *-----------------------------------------------------------------------*/

#define MHASH(s,size)   ((*s)%size)

/*
 * 1つの音節に対する名前、番号（各音節には順番に番号が付けられている）
 * を格納する構造体。
 */
typedef struct _entry
{
        char    *key;           /* 音節名 */
        int     data;           /* 音節の番号 */
        struct _entry   *next;  /* 次の entry へのポインタ */
}       entry;

/*
 * ハッシュ・テーブルを表す構造体。
 */
typedef struct _hashtable
{
        entry   **entrytable;   /* 構造体 entry を指す */
        int     size;           /* ハッシュ・テーブルの大きさ */
}       hashtable;

hashtable       *init_hash();
```

LINE # HEADER TEXT

```
/*-----------------------------------------------------------------------
 *
 *      大域変数の定義
 *
 *          ATR Interpreting Telephony Research Laboratories.
 *
 *      History
 *
 *          9-Jul-1990  Created by Kenji Kita and Toshiyuki Hanazawa.
 *
 *-----------------------------------------------------------------------*/

#include "HMM.h"
#include "LR.h"
#include "cell.h"

/*
 * HMMのパラメータを読み込むのは時間がかかるため、パラメータを読み込んだ
 * 時点でのプログラムの実行イメージを作っておくと便利がよい。
 * 変数 initialized がセットされていれば、既に、HMM のパラメータが
 * ロードされていることを示す。
 * 関数 save_image() でセットされる。
 */
#ifdef HMMLR_MAIN
  int   initialized = 0;
#else
  int   initialized;
#endif

#ifndef HMMLR_MAIN
extern
#endif
int     dynamicthreshold,       /* 音韻照合の threshold を動的に変えるか？ */
        Nbest,                  /* 何個の候補候補を出力するか？ */
        Verbose,                /* 冗長なモード */
        Trace,                  /* 現在、未使用 */
        Trace2,                 /* 現在、未使用 */
        First,                  /* 花沢氏作成のオプションであるが、
                                   First ではなくて、Fast の間違いであると思う。
                                   関数 cut_prob() を参照のこと */
        Lmodel,                 /* 統計的言語モデルを使うか？ */
        int_threshold,          /* threshold (integer) */
        int_minimalthreshold,   /* threshold (integer) の下限 */
        total_verify,           /* 駆動された音韻照合の数 */
        NormProb[MAXFRAME];     /* 正規化確率保存用の配列 */

#ifndef HMMLR_MAIN
extern
#endif
double  threshold,              /* threshold */
        threshold_0,            /* 最初の無音に対する threshold */
        originalthreshold,      /* 最初の threshold */
        minimalthreshold,       /* threshold の下限 */
        log_henkan,
        log_henkan10;

#ifdef HMMLR_MAIN
  double        W_syllable = 0.0;       /* 現在、未使用 */
#else
  double        W_syllable;
#endif

#ifndef HMMLR_MAIN
extern
#endif
char    Hmmlist[80],            /* HMM音韻モデルおよび継続時間長
                                   パラメータを指定するファイル名 */
        Logtable[80];           /* log compression table ファイル */

/*
 * 文法および LR解析表。
 * Gtable, Atable は、2段階LR使用時の文節間文法のためのもの。
 */
#ifndef HMMLR_MAIN
extern
#endif
Rule    *GrammarTable[MAXGRAMMARS],
        *Gtable[MAXGRAMMARS];

#ifndef HMMLR_MAIN
extern
#endif
Action  *ActionTable[MAXSTATES],
        *Atable[MAXSTATES];

/*
 * 話者適応用のヒストグラム。
 */
#ifndef HMMLR_MAIN
extern
#endif
int     Tophist;

#ifndef HMMLR_MAIN
extern
#endif
char    Histlist[80];

/*
 * 入力ファイルへのポインタ。
 */
#ifndef HMMLR_MAIN
extern
#endif
FILE    *labelfp;

/*
 * 関数の宣言。
 */
double  log(), rint();
long    cputime(), realtime();
```

LINE # | HEADER TEXT

```
1  #define  FLAT       (-1)
2
3
4  struct iduration
5  {
6    int      states;
7    int      lenmin,
8             lenmax,
9             lenmin2,
10            lenmax2,
11            *lenprob;
12   double   lenav,
13            lensd;
14   int      durmin[MAX_STATES],
15            durmax[MAX_STATES],
16            *durprob[MAX_STATES];
17 };
18
19 typedef  struct iduration       IDUR;
```

LINE # | HEADER TEXT

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define TRUE                    1
5  #define FALSE                   0
6  #define MIN_INT                 (-111111111)
7  #define MAX_ALPHABET            256
8  #define MAX_STATES              15
9  #define MAX_ARCS                45
10 #define NULL_TRANSITION         (-1)
11 #define NO_ROOP                 (-2)
12
13 #define MAX_CODEBOOK            3
14
15
16 struct transition
17 {
18   int     destination,
19           origin,
20           out_prob_index;
21   int     trans_prob;
22 };
23
24
25 struct state
26 {
27   int   label,
28         num_from,
29         num_to,
30         is_initial,
31         is_final,
32         trans_from[MAX_STATES],  /* All transitions from it */
33         trans_to[MAX_STATES],    /* All transitions to   it */
34         rarc;
35 };
36
37
38 struct imhmm
39 {
40   int                 num_states,
41                       num_omatrix,
42                       num_arcs,
43                       num_initial,
44                       num_final,
45                       num_book,
46                       num_vector[MAX_CODEBOOK];
47   float               *output_prob[MAX_CODEBOOK];
48   int                 *frame_prob[MAX_ARCS];
49   struct state        *states;
50   struct transition   *transitions;
51 };
52
53 typedef  struct imhmm           IMHMM;
54 typedef  struct training_set    TSET;
```

LINE # | HEADER TEXT

```
1  #define logx( a, b )            ( log(b) / log(a) )
2
3  struct log_table
4  {
5    double  b;
6    int     size,
7            *table;
8  };
9
10 typedef  struct log_table       LGTBL;
11
12 #ifndef  LOG_MAIN
13 extern
14 #endif
15 LGTBL  Lgtbl;
```

LINE # | HEADER TEXT

```
/*-----------------------------------------------------------------------
 *
 *      N-gram データ格納用構造体の定義
 *
 *          ATR Interpreting Telephony Research Laboratories.
 *
 *      History
 *
 *          27-Oct-1989  Created by Kenji Kita.
 *
 *-----------------------------------------------------------------------*/

#define HASH(n,size)    (n % size)

/*
 * シンボルのＮ組データ１つを格納する構造体．
 * シンボルが m 個あるとき，これら m 個のシンボルに順番に番号を振る．
 * そして，Ｎ組データは，これらの番号を使って n 進法で表現する．
 */
typedef struct _glist
{
        unsigned int    key;    /* n 進法で表したＮ組データの値 */
        float   prob;           /* Ｎ組データの個数 */
        struct _glist   *next;  /* 次のＮ組データへのポインタ */
} glist;

/*
 * 文字列のＮ組データを格納するテーブル．
 */
typedef struct _gtable
{
        glist   **table;        /* glist（上で定義）へのポインタ */
        int     tablesize;      /* table の大きさ */
        int     ngram;          /* 何個組のデータか？ */
        int     nsymbols;       /* シンボルの総数 */
        char    **symbol;       /* シンボル・テーブル */
} gtable;
```

LINE # SOURCE TEXT

```
  1 /*-----------------------------------------------------------------------
  2  *
  3  *    HMM-LRのメイン・プログラム
  4  *
  5  *    オプション解析、初期化、認識のメイン・ループを含む。
  6  *
  7  *    ソース・ファイルのあるディレクトリ
  8  *
  9  *      atr-dp:/ifg2/LR/HMM-LR/Hmmlr.V7J
 10  *
 11  *
 12  *          ＡＴＲ自動翻訳電話研究所
 13  *          (ATR Interpreting Telephony Research Laboratories)
 14  *
 15  *          ＬＲ作成担当:  北　研二 (Kenji Kita)
 16  *          ＨＭＭ作成担当: 花沢 利行 (Toshiyuki Hanazawa)
 17  *          日本語コメント: 北　研二 (Kenji Kita)
 18  *
 19  *
 20  *    注意
 21  *
 22  *      (１) ＨＭＭ-ＬＲのプログラムの全部または一部を無断で
 23  *           使用しないで下さい。
 24  *      (２) 本プログラムのコードは、将来予告なしに変更することがあります。
 25  *      (３) 本プログラムの内容に関して、万一御不審な点、バグなど
 26  *           お気づきの点がございましたら、作成者にお申しつけください。
 27  *           ただし、即答出来にはそいかねる場合もあります。
 28  *      (４) 本プログラムを運用した結果の影響につきましては、
 29  *           (３)の項にかかわらず、責任を負い兼ねますので、
 30  *           御了承ください。
 31  *      (５) 本プログラム中からコメントを削除しないでください。
 32  *           コメントを付けるのには、多大な労力と時間がかかっています。
 33  *
 34  *
 35  *    ＨＭＭ-ＬＲの開発の歴史
 36  *
 37  *      Version 0.0  25-Jun-1988.  HMM-LR prototype. [kita]
 38  *      Version 1.0  25-May-1989.  Use separate VQ. [kita & hanazawa]
 39  *      Version 2.0  13-Sep-1989.  2nd version. [hanazawa]
 40  *      Version 3.0  29-Sep-1989.  3rd version. [hanazawa]
 41  *      Version 4.0  12-Oct-1989.  4th version. [hanazawa]
 42  *      Version 5.0  22-Feb-1990.  Fast beam-search. [kita]
 43  *      Version 5.5  07-Mar-1990.  [kita]
 44  *      Version 6.0  08-Mar-1990.  Two-level LR. [kita]
 45  *      Version 6.1  20-Jun-1990.  DEC_TO_IEEE. [kita]
 46  *      Version 6.2   3-Jul-1990.  [kita]
 47  *      Version 6.3   9-Jul-1990.  syllable trigram. [kita]
 48  *      Version 6.4  29-Jun-1991.  LR-CRT. [kita]
 49  *      Version 6.5   5-Jul-1991.  Best-first search. [kita]
 50  *      Version 7.0  18-Feb-1992.  プログラムの整理、
 51  *                                 日本語のコメント付与. [北 研二]
 52  *
 53  *-----------------------------------------------------------------------*/
 54
 55 /*
 56  * 使い方の例
 57  *
 58  * % Hmmlr /ifg1/K-TH-FZY/mau_sbj_fzy.wd \
 59  *       -g /ifg2/LR/Grammar/sp50 \
 60  *       -m /ifg1/MAU-TH/HMM/LIST-DEVOC1/List_9mQ81Ddceps_nosmthUJ \
 61  *       -P 9 -M 3 -B 100 -b 18 -c 1000 -s 2.0 -W 1.0 \
 62  *       -l @G 0.04
 63  *
 64  * 2段階ＬＲの使い方の例
 65  *
 66  * % Hmmlr /ifg2/LR/HMM-LR/TwoLevel/LBL/MODELCONV.LBL \
 67  *       -g /ifg2/LR/HMM-LR/TwoLevel/GRA/dp10-re72-new.two-level \
 68  *       -G /ifg2/LR/HMM-LR/TwoLevel/GRA/bun7 \
 69  *       -m /ifg1/MAU-TH/HMM/LIST-DEVOC1/List_9mQ81Ddceps_nosmthUJ \
 70  *       -P 9 -M 3 -B 100 -b 18 -c 1000 -s 2.0 -W 1.0
 71  */
 72
 73 /*
 74  * ＨＭＭ－ＬＲのバージョン番号。
 75  */
 76 char    *Version = "V7";
 77
 78 /*
 79  * Copyright 削除しないこと。
 80  */
 81 char    *Copyright = "Copyright(C) 1988,1989,1990,1991,1992  K.Kita & T.Hanazawa";
 82
 83 /*
 84  * このプログラムが、メイン・プログラムである。
 85  */
 86 #define HMMLR_MAIN
 87
 88 #include <stdio.h>
 89 #include <math.h>
 90 #include "config.h"
 91 #include "hmmlr.h"
 92 #include "log.h"
 93
 94 char    labelfile[80],          /* ＶＱコード・インデックス・ファイル */
 95         progname[80],           /* HMM-LR実行ファイル名 */
 96         dumpfile[80],           /* 数行 下を参照 */
 97         dumpdate[80],           /*      同　上     */
 98         GrammarName[80],        /* 文法名 */
 99         GrammarName2[80];       /* ２段階HMM-LRで用いる文節間文法の名前 */
100
101 /*
102  * 変数 dumpfile, dumpdate の説明
103  *
104  * HMMのパラメータを読み込むのは時間がかかるため、パラメータを読み込んだ
105  * 時点でのプログラムの実行イメージを作っておくと便利がよい。
106  * 変数 dumpfile は、このような実行イメージを保存しておくためのファイルである。
107  * また、dumpdate は dumpfile が作られた日時を保存するための変数である。
108  * 本ファイルの最後の関数 save_image が、この機能を実現する関数である。
109  */
110
111 /*
112  * プログラムの実行時オプション一覧を表示する。
113  * 各オプションの意味については、main() 関数中の
114  * オプション解析の箇所を参照のこと。
115  */
116 Usage()
117 {
118   fprintf(stderr, "Usage: %s VQ-labelfile (with following options)\n",
119           progname);
120   fprintf(stderr, "  -G grammar(inter-phrase)\n");
```

LINE # SOURCE TEXT

```
121    fprintf(stderr, "  -g grammar\n");
122    fprintf(stderr, "  -B global-beam\n");
123    fprintf(stderr, "  -b local-beam\n");
124    fprintf(stderr, "  -c total-cells\n");
125    fprintf(stderr, "  -X Threshold\n");
126    fprintf(stderr, "  -x Minimal-threshold\n");
127    fprintf(stderr, "  -Q Threshold_Q\n");
128    fprintf(stderr, "  -d (Disable dynamic threshold setting)\n");
129    fprintf(stderr, "  -n N-best\n");
130    fprintf(stderr, "  -m hmm-list\n");
131    fprintf(stderr, "  -L log-table\n");
132    fprintf(stderr, "  -S n*sigma (model-duration-window)\n");
133    fprintf(stderr, "  -s n*sigma (state-duration-window)\n");
134    fprintf(stderr, "  -N n*state-standard-deviation\n");
135    fprintf(stderr, "  -W duration-weight\n");
136    fprintf(stderr, "  -w syllable-weight\n");
137    fprintf(stderr, "  -V (Verbose)\n");
138    fprintf(stderr, "  -T (Trace)\n");
139    fprintf(stderr, "  -t (Trace bun)\n");
140    fprintf(stderr, "  -F (First-trellis)\n");
141    fprintf(stderr, "  -P frame-period (msec)\n");
142    fprintf(stderr, "  -A add-time (msec MAE & USHIRO)\n");
143    fprintf(stderr, "  -E end-free (msec)\n");
144    fprintf(stderr, "  -M mabiki(1=3msec, 3=9msec)\n");
145    fprintf(stderr, "  -H adaptation histgram\n");
146    fprintf(stderr, "  -o dump-file\n");
147    fprintf(stderr, "  -l (Use stochastic lmodel)\n");
148    exit(1);
149  }
150
151  /*
152   * 実行時の各種パラメータの値を表示する。
153   */
154  OpenMsg()
155  {
156    printf("    [ HMM-LR Continuous Speech Recognizer. %s ]\n", Version);
157    printf("    <<< %s >>>\n", Copyright);
158
159  #if TWO_LEVEL
160    printf("\n ***** TWO-LEVEL *****\n");
161  #endif
162  #if BEST_FIRST
163    printf("\n ***** BEST-FIRST SEARCH *****\n");
164  #endif
165
166    printf("\n ### %s (%s) ###\n", progname, Version);
167    if(initialized) printf(" Created on %s\n", dumpdate);
168    if(TRELLIS)           printf("\n Trellis algorithm && ");
169    else                  printf("\n Viterbi algorithm && ");
170    if(DURATION_CONTROL)  printf("Duration control.\n\n");
171    else                  printf("No duration.\n\n");
172
173    printf(" Grammar-> %s\n", GrammarName);
174    if(*GrammarName2) printf(" Grammar(bun)-> %s\n\n", GrammarName2);
175    printf(" Total cells = %d.\n", totalcells);
176    printf(" ThresholdQ = %f.\n", threshold_Q);
177    printf(" Threshold1 = %f.\n", originalthreshold);
178    printf(" Threshold2 = %f.\n", minimalthreshold);
179    printf(" Global beam = %d.\n", globalbeam);
180    printf(" Local beam = %d.\n", localbeam);
181    printf(" First = %d\n\n", First);
182    printf(" Input-> %s\n", labelfile);
183    printf(" Frame period = %3.1f msec\n", Frame_Period);
184    printf(" Add time = %4.1f msec\n", Add_Time);
185    printf(" End_Free = %4.1f msec\n", End_Free);
186    printf(" Mabiki = %d\n\n", Mabiki);
187    fflush(stdout);
188  }
189
190  /*
191   * オプション指定の後に、代入する値があるかどうかをチェックする。
192   */
193  #define CheckOption(opt)  if(argc - 1 <= 0 || **(argv+1) == '-')\
194                            fatal_error("! Missing value after %s option\n",opt)
195
196  /*
197   * 最初の無音照合用の threshold（デフォルト値）。
198   */
199  #define THRESHOLD_Q             30.0
200
201  /*
202   * 音韻照合用の threshold（デフォルト値）。
203   */
204  #define ORIGINAL_THRESHOLD      20.0
205
206  /*
207   * 音韻照合用の threshold の下限（デフォルト値）。
208   * dynamicthreshold という変数がセットされていると、
209   * 関数 hmmlr() の中で、threshold の値は徐々に小さな値に変更される。
210   * このときの threshold の下限を示す。
211   */
212  #define MINIMAL_THRESHOLD       15.0
213
214  main(argc, argv)
215  int     argc;
216  char    *argv[];
217  {
218          double  atof();
219          Entry   *ep,            /* 音韻モデルを格納している EntryTable
220                                     へのポインタ */
221                  *lookup();      /* 音韻名から EntryTable へのポインタ
222                                     を得るための関数 */
223          double  durminQ,        /* 無音区間の最小長 */
224                  durmaxQ;        /* 無音区間の最大長 */
225
226          /*
227           * 実行プログラムの名前を変数 progname に代入。
228           */
229          strncpy(progname, argv[0], sizeof(progname));
230
231          /*
232           * プログラムの引数が全くなければ、オプションの一覧を表示。
233           */
234          if(argc < 2) Usage();
235
236          /*
237           * 各種パラメータの初期化。
238           */
239          *labelfile = NULL;
240          *dumpfile = NULL;
```

```
241         *GrammarName = NULL;
242         *GrammarName2 = NULL;
243
244         *Hmmlist = NULL;
245         *Logtable = NULL;
246         *Histlist = NULL;
247
248         dynamicthreshold = 1;
249         Nbest = 5;
250         Verbose = 0;
251         Trace = 0;
252         Trace2 = 0;
253         First = 0;
254         Lmodel = 0;
255
256         /******* cells *******/
257
258         totalcells = TOTALCELLS;
259         globalbeam = GLOBAL_BEAM;
260         localbeam = LOCAL_BEAM;
261
262         /******* threshold *******/
263
264         threshold_0 = THRESHOLD_0;
265         originalthreshold = ORIGINAL_THRESHOLD;
266         minimalthreshold = MINIMAL_THRESHOLD;
267
268         /******* duration control parameter *******/
269
270         Sigma_model = 3.0; Sigma_state = 3.0; Nsd = 1.0; Dur_weight = 7.0;
271
272         /******* frame period *******/
273
274         Frame_Period = FRAME_PERIOD;
275
276         Add_Time = ADD_TIME;
277         End_Free = END_FREE;
278
279         Mabiki = 1;
280
281         /******* adaptation histgram *******/
282
283         Tophist = 20;
284
285         /*
286           * オプションの解析
287           * オプションが非常に沢山あってみにくいが、
288           * 初期のバージョンでは少数のオプションしかなかった。
289           */
290         while(--argc > 0)
291         {
292             if(**++argv == '-')
293             {
294                 switch((*argv)[1])
295                 {
296                 case 'g':  /* 文法名 */
297                     CheckOption("g");
298                     strncpy(GrammarName, *(argv+1), sizeof(GrammarName));
299                     --argc;
300                     ++argv;
301                     break;
302                 case 'G':  /* 2段階LRの場合の文節間文法名 */
303                     CheckOption("G");
304                     strncpy(GrammarName2, *(argv+1), sizeof(GrammarName2));
305                     --argc;
306                     ++argv;
307                     break;
308                 case 'B':  /* ビーム幅 */
309                     CheckOption("B");
310                     globalbeam = atoi(*(argv+1));
311                     --argc;
312                     ++argv;
313                     break;
314                 case 'b':  /* 局所的なビーム幅
315                               (1つの仮説からの最大分岐数 */
316                     CheckOption("b");
317                     localbeam = atoi(*(argv+1));
318                     --argc;
319                     ++argv;
320                     break;
321                 case 'c':  /* 確保するセル数 */
322                     CheckOption("c");
323                     totalcells = atoi(*(argv+1));
324                     --argc;
325                     ++argv;
326                     break;
327                 case 'X':  /* threshold */
328                     CheckOption("X");
329                     originalthreshold = atof(*(argv+1));
330                     --argc;
331                     ++argv;
332                     break;
333                 case 'x':  /* threshold の下限 */
334                     CheckOption("x");
335                     minimalthreshold = atof(*(argv+1));
336                     --argc;
337                     ++argv;
338                     break;
339                 case 'Q':  /* 最初の無音に対する threshold */
340                     CheckOption("Q");
341                     threshold_0 = atof(*(argv+1));
342                     --argc;
343                     ++argv;
344                     break;
345                 case 'd':  /* threshold を自動的に再設定するか？ */
346                     dynamicthreshold = 0;
347                     break;
348                 case 'n':  /* 最終的に何個の候補を出力するか？ */
349                     CheckOption("n");
350                     Nbest = atoi(*(argv+1));
351                     --argc;
352                     ++argv;
353                     break;
354                 case 'm':  /* HMM音韻モデルおよび継続時間長
355                               パラメータを指定するファイル名 */
356                     CheckOption("m");
357                     strcpy(Hmmlist, *(argv+1));
358                     --argc;
359                     ++argv;
360                     break;
```

LINE # | SOURCE TEXT

```
361                 case 'L':  /* log compression table ファイル名 */
362                     CheckOption("L");
363                     strcpy(Logtable, *(argv+1));
364                     --argc;
365                     ++argv;
366                     break;
367                 case 'S':  /* 花沢コード */
368                     CheckOption("S");
369                     Sigma_model = atof(*(argv+1));
370                     --argc;
371                     ++argv;
372                     break;
373                 case 's':  /* 花沢コード */
374                     CheckOption("s");
375                     Sigma_state = atof(*(argv+1));
376                     --argc;
377                     ++argv;
378                     break;
379                 case 'N':  /* 花沢コード */
380                     CheckOption("N");
381                     Nsd = atof(*(argv+1));
382                     --argc;
383                     ++argv;
384                     break;
385                 case 'W':  /* 継続時間長制御に対する重み */
386                     CheckOption("W");
387                     Dur_weight = atof(*(argv+1));
388                     --argc;
389                     ++argv;
390                     break;
391                 case 'w':  /* 現在 未使用 */
392                     CheckOption("w");
393                     W_syllable = atof(*(argv+1));
394                     --argc;
395                     ++argv;
396                     break;
397                 case 'V':  /* 冗長なモード */
398                     Verbose = 1;
399                     break;
400                 case 'T':  /* 現在 未使用 */
401                     Trace = 1;
402                     break;
403                 case 't':  /* 現在 未使用 */
404                     Trace2 = 1;
405                     break;
406                 case 'F':  /* 花沢氏作成のオプションであるが
407                               First ではなくて Past の間違いであると思う.
408                               関数 cut_prob() を参照のこと */
409                     First = 1;
410                     break;
411                 case 'P':  /* フレーム長 */
412                     CheckOption("P");
413                     Frame_Period = atof(*(argv+1));
414                     --argc;
415                     ++argv;
416                     break;
417                 case 'E':  /* 関数 hmmlr() を参照のこと */
418                     CheckOption("E");
419                     End_Free = atof(*(argv+1));
420                     --argc;
421                     ++argv;
422                     break;
423                 case 'A':  /* 音声データを読み込む際に 前後に
424                               Add_Time 分の無音区間を含める */
425                     CheckOption("A");
426                     Add_Time = atof(*(argv+1));
427                     --argc;
428                     ++argv;
429                     break;
430                 case 'M':  /* 音声データに対する間引きを示す */
431                     CheckOption("M");
432                     Mabiki = atoi(*(argv+1));
433                     --argc;
434                     ++argv;
435                     break;
436                 case 'H':  /* 話者適応用ヒストグラム ファイル */
437                     CheckOption("H");
438                     strcpy(Histlist, *(argv+1));
439                     --argc;
440                     ++argv;
441                     break;
442                 case 'o':  /* 変数 dumpfile 参照のこと */
443                     CheckOption("o");
444                     strncpy(dumpfile, *(argv+1), sizeof(dumpfile));
445                     --argc;
446                     ++argv;
447                     break;
448                 case 'l':  /* 統計的言語モデルを用いるか？ */
449                     Lmodel = 1;
450                     break;
451                 default:  /* オプション指定の間違い */
452                     fatal_error("! Unknown option : %c\n", (*argv)[1]);
453                 }
454             }
455             /*
456              * @ の後は 統計的言語モデルの重みを指定する.
457              * あまりにも オプション指定の数が多くなったので,
458              * 仕方なく新しいオプション指定子 @ を作った.
459              */
460             else if(**argv == '@')
461             {
462                 extern double   W1, W2;
463                 switch((*argv)[1])
464                 {
465                 case 'G':  /* 確率文法に対する重み */
466                     CheckOption("@G");
467                     W1 = atof(*(argv+1));
468                     --argc;
469                     ++argv;
470                     break;
471                 case 'S':  /* 音節 trigram に対する重み */
472                     CheckOption("@S");
473                     W2 = atof(*(argv+1));
474                     --argc;
475                     ++argv;
476                     break;
477                 default:  /* オプション指定の間違い */
478                     fatal_error("! Unknown option @: %c\n", (*argv)[1]);
479                 }
480             }
```

LINE # SOURCE TEXT

```
481             else  /* VQラベル・ファイル名 */
482                 strcpy(labelfile, *argv);
483         }
484
485         /*
486          * 入力ファイルが指定されていなければ困る。
487          */
488         if(*labelfile == NULL)
489                 fatal_error("Missing label file\n");
490
491         /*
492          * totalcells の値に比べて、globalbeam の値は十分大きいこと。
493          * とりあえず、totalcells は、globalbeam の３倍以上と
494          * しているが、これは別に根拠なし。
495          */
496         if(globalbeam > totalcells/3)
497                 fatal_error("Too large beam.\n");
498
499         /*
500          * 実行時パラメータの値を表示。
501          */
502         OpenMsg();
503
504         /*
505          * HMM-LRの初期化を行なう。
506          */
507         init();
508
509         /*
510          * 無音区間の長さの許容範囲を調べて、表示する。
511          */
512         op = lookup("Q2");
513         durminQ = (double)(ep->duration[0]->lenmin) * Frame_Period;
514         durmaxQ = (double)(ep->duration[0]->lenmax) * Frame_Period;
515         printf(" DurationQ: %4.1f - %4.1f msec\n\n", durminQ, durmaxQ);
516         fflush(stdout);
517
518         log_henkan = log(Lgtbl.b);
519         log_henkan10 = log10(Lgtbl.b);
520
521         /*
522          * 入力音声ファイルをオープンする。
523          */
524         if((labelfp = fopen(labelfile, "r")) == NULL)
525                 fatal_error("Can't open %s\n", labelfile);
526
527         /*
528          * GrammarName2（文節間文法）が与えられているときには、
529          * 文節間LRパーサを呼び出す。
530          */
531         if(*GrammarName2)
532             LRctrl_main(labelfp);
533         else
534         /*
535          * GrammarName2が指定されていないときは、以下のwhileループを繰り返す。
536          */
537         while(1)
538         {
539                 int     st;
540
541                 /*
542                  * 音声データを読み込む。
543                  */
544                 if((st = Read_Fuzzy(labelfp)) == EOF)
545                         break;
546                 else if(st == FALSE)
547                         continue;
548
549                 /*
550                  * log likelihood map の作成。
551                  */
552                 cal_frame_prob();
553
554                 /*
555                  * セル領域の初期化。
556                  */
557                 reinitcell();
558
559                 /*
560                  * 音声認識。
561                  */
562                 hmmlr();
563         }
564         /*
565          * 入力ファイルをクローズし、処理を終了する。
566          */
567         fclose(labelfp);
568 }
569
570 /*
571  * HMM-LRの初期化を行なう。
572  */
573 init()
574 {
575         long    csec, rsec, cputime(), realtime();
576
577         /*
578          * タイマの設定。初期化に要する時間を計るため。
579          */
580         cputime();
581         realtime();
582
583         /*
584          * HMMのモデルのパラメータ、継続時間長パラメータを読み込む。
585          * initialized という変数がセットされていれば、すでにHMMの
586          * パラメータは読み込まれている。（関数 save_image 参照）
587          */
588         if(!initialized)
589         {
590                 init_HMM();
591
592                 /*
593                  * もし dumpfile が与えられていれば、
594                  * この時点で実行イメージを書き出す。
595                  */
596                 if(*dumpfile)
597                 {
598                         save_image();
599                         exit(1);
600                 }
```

```
LINE #  SOURCE TEXT
601             }
602
603     #if TWO_LEVEL
604             /*
605              * 文節内文法およびLR解析表を読み込む．
606              */
607             ReadLR2(1, GrammarName, GrammarTable, ActionTable);
608     #else
609             /*
610              * 文法およびLR解析表を読み込む．
611              */
612             ReadLR(1, GrammarName, GrammarTable, ActionTable);
613     #endif
614             /*
615              * GrammarName2 が指定されていれば，
616              * 文節間文法およびLR解析表を読み込む．
617              */
618             if(*GrammarName2)
619                     ReadLR(1, GrammarName2, Gtable, Atable);
620
621             /*
622              * セル領域を確保する．
623              */
624             init_cell();
625
626             /*
627              * 統計的言語モデルの scaling parameter 設定．
628              */
629             if(Lmodel) init_scaling();
630
631     #if SYLLABLE_NGRAM
632             /*
633              * 音節の trigram データを読み込む．
634              */
635             if(Lmodel) init_sngram();
636     #endif
637
638             /*
639              * ビームサーチ用のバッファを確保．
640              */
641             init_beam();
642
643             /*
644              * 初期化終了．
645              */
646             csec = cputime();
647             rsec = realtime();
648
649             fprintf(stderr,
650               "Initializing complete!! CPU-time=%ld msec., Elapsed-time=%ld sec.\n",
651               csec, rsec);
652             return(TRUE);
653     }
654
655     /*
656      * プログラムの実行イメージを dumpfile という名前のファイルに書き出す．
657      * 諸般の事情により，本バージョンではこの機能は削除（コメント）してある．
658      */
659     save_image()
660     {
661     /*
662             extern int      etext, edata;
663             long    time(), tt;
664             char    *ctime();
665
666             time(&tt);
667             strcpy(dumpdate, ctime(&tt));
668
669             fprintf(stderr, "--> Dumping image to `%s'...\n", dumpfile);
670             fflush(stderr);
671             ++initialized;
672             unexec(dumpfile, progname, etext, sbrk(0), 0);
673     */
674     }
```

LINE # SOURCE TEXT

```
  1 /*---------------------------------------------------------------------*
  2  *                                                                     *
  3  *      ＨＭＭ-ＬＲの音声認識部                                       *
  4  *                                                                     *
  5  *          ＡＴＲ自動翻訳電話研究所                                   *
  6  *          (ATR Interpreting Telephony Research Laboratories)         *
  7  *                                                                     *
  8  *          ＬＲ作成担当：  北　研二 (Kenji Kita)                      *
  9  *          ＨＭＭ作成担当：花沢 利行 (Toshiyuki Hanazawa)             *
 10  *          日本語コメント：北　研二 (Kenji Kita)                      *
 11  *                                                                     *
 12  *      History                                                        *
 13  *                                                                     *
 14  *          29-Jun-1991, Two-level version;                            *
 15  *          5-Jul-1991, Best-first version;                            *
 16  *          5-Dec-1991, Best-first version modified.                   *
 17  *                                                                     *
 18  *---------------------------------------------------------------------*/
 19
 20 #include <stdio.h>
 21 #include <math.h>
 22 #include "config.h"
 23 #include "hmmlr.h"
 24 #include "log.h"
 25
 26 /*
 27  * streq(s,t) : 文字列 s と t が等しいかを判断する。
 28  */
 29 #define streq(s,t)      (*(s) == *(t) && strcmp(s,t) == 0)
 30
 31 /*
 32  * MIN(x,y) : x と y のうち、小さい方の値。
 33  */
 34 #define MIN(x,y)        (((x) < (y)) ? (x) : (y))
 35
 36 /*
 37  * MAX(x,y) : x と y のうち、大きい方の値。
 38  */
 39 #define MAX(x,y)        (((x) < (y)) ? (y) : (x))
 40
 41 #define MORA_LEN        100.0
 42
 43 #if TWO_LEVEL
 44 /*
 45  * ２段階ＬＲで、文節間ＬＲが予測した文節カテゴリを保持する。
 46  * topcat[i] が０でなければ、文節カテゴリ SymbolTable[topcat[i]]
 47  * が予測されていることになる。
 48  */
 49 int     topcat[MAXSYMBOLS];
 50 #endif
 51
 52 /*
 53  * 音声認識を行なう、ＨＭＭ-ＬＲの主要部。
 54  */
 55 hmmlr()
 56 {
 57         register int    i;      /* general use */
 58         register char   *p, *q; /* general use */
 59         register Cell   *cp,    /* 現在の仮説セルへのポインタ */
 60                         *newcp, /* 新しい仮説セルへのポインタ */
 61                         *cpp;   /* general use */
 62         int     m,              /* 還元動作時の、文法規則の右辺の長さ */
 63                 count,          /* 仮説の総数をカウントする */
 64                 state,          /* LR解析表の現在の状態 */
 65                 newstate,       /* GOTO 関数の行き先 */
 66                 action,         /* 現在のLR動作 */
 67                 offs,           /* １つの仮説から分岐する仮説数をカウント */
 68                 st;             /* 音韻照合が成功したかどうかを示すフラグ */
 69         char    phone[16],      /* 照合する音韻名 */
 70                 phone2[16];     /* 照合する音韻名（最後の母音）*/
 71         Action  *ap;            /* LR解析表の動作項を走査 */
 72         Cell    *getcopy1(),    /* ファイル cell.c 参照 */
 73                 *getcopy2();    /*      同　上      */
 74         int     int_tx,         /* threshold の再設定時に一時的に使用 */
 75                 other_action,   /* LR解析表の状態に還元動作以外の動作が
 76                                    あるかどうかを覚えておく */
 77                 flg_vow,        /* 母音に対しては、２種類の継続時間長
 78                                    パラメータがあるが、このうちの
 79                                    どちらを用いるかを示すフラグ */
 80                 end_free;       /* end_free を初期化している部分参照のこと */
 81         int     Depth,          /* 認識の深さ（仮説中の音韻数）*/
 82                 Max_Depth;      /* 認識の深さの上限 */
 83         float   Data_Time;      /* 入力音声長 */
 84         char    buf[BUFSIZ];    /* 認識された音韻列 */
 85 #if BEST_FIRST
 86         int     acc_count = 0,  /* 受理された仮説セルの数 */
 87                 acc_flag;       /* 新たな音韻照合で受理されたセルが
 88                                    あるかどうかを覚えておくために使用 */
 89         int     top_prob;       /* 最良の尤度 */
 90         Cell    *top_cell;      /* 最良の仮説セル */
 91 #endif
 92
 93         /*
 94          * 以下の３行は、花沢コード。
 95          * 入力音声長から、だいたい何音韻くらいの長さまでを照合すれば
 96          * よいか決定している。
 97          * 認識された音韻木の深さ Depth が Max_Depth よりも深くなれば、
 98          * その時点で認識を止めるようになっている。
 99          */
100         Data_Time = (float)N_frames * (float)Frame_Period;
101         Max_Depth = (int)(Data_Time / MORA_LEN * 2.0) + 2;
102         Max_Depth = MIN(25, Max_Depth);
103                 /*      ↑
104                  * この値に注意。この値は DSDJ 用のもの。
105                  * この値では、25音韻以上の音声は認識されない。
106                  */
107
108         /*
109          * 音声の最初には無音が含まれていると仮定しているので、
110          * threshold を無音用のものに設定する。
111          */
112         threshold = threshold_0;
113         int_threshold = (int)(threshold / log_henkan);
114         int_minimalthreshold = (int)(minimalthreshold / log_henkan);
115
116         /*
117          * HMM音韻照合が何回駆動されたかをカウントする変数の初期化。
118          * total_verify は、関数 verify の中でインクリメントされる。
119          */
120         total_verify = 0;
```

```
LINE #  SOURCE TEXT
  121
  122           /*
  123            * 認識の深さ（音韻数）を記憶する変数の初期化．
  124            */
  125           Depth = 0;
  126
  127           /*
  128            * 変数 end_free を初期化する．
  129            * 最終的な認識候補の尤度は，音声の終端から end_free フレーム
  130            * 以内にある各フレームの正規化尤度のうち，最もよいものが採用される．
  131            */
  132           end_free = (int)(End_Free / Frame_Period);
  133
  134           /*
  135            * 認識時間を測定するために，クイマの設定を行なう．
  136            */
  137           cputime();
  138           realtime();
  139
  140           /*
  141            * 入力音声の最初の無音の照合．
  142            */
  143           verify_start(celltop, -(int_threshold));
  144
  145   #if BEST_FIRST
  146           /*
  147            * Best-first 探索を行なう場合は，一番尤度の高い仮説に対してだけ，
  148            * 新たな音韻照合が行なわれる．
  149            * 新たな音韻照合が行なわれた仮説セルでは，変数 shifted が 1 にセットされる．
  150            * 認識動作を始めるに当たり，この変数をセットしておく．
  151            * 変数 shifted がセットされていないセルに対しては，
  152            * 還元動作を行なわない．
  153            */
  154           celltop->shifted = 1;
  155   #endif
  156
  157           /*
  158            * 認識のメイン・ループ．
  159            * ループ 1 回ごとに，各認識仮説に対し，新たな音韻が認識される．
  160            */
  161           while(1)
  162           {
  163   #if !BEST_FIRST
  164               /*
  165                * 認識の深さ（音韻数）Depth が，Max_Depth よりも
  166                * 大きくなれば，認識処理を止める．
  167                */
  168               if(++Depth >= Max_Depth)
  169                   break;
  170   #endif
  171               /*
  172                * 以下の while ループで，LR パーザの還元動作を行なう．
  173                */
  174               while(celltop)
  175               {
  176                   /*
  177                    * celltop の先頭のセル cp を取り出す．
  178                    * cp に対し，還元動作を行なう．
  179                    */
  180                   cp = celltop;
  181                   celltop = celltop->next;
  182
  183                   /*
  184                    * 還元動作以外の動作があるかを覚えておくための変数の初期化．
  185                    */
  186                   other_action = 0;
  187
  188                   /*
  189                    * LR解析表の現在の状態を state に代入．
  190                    */
  191                   state = stacktop(cp);
  192   #if BEST_FIRST
  193                   /*
  194                    * 変数 shifted がセットされていない仮説セルは，
  195                    * 新たな移動動作（即ち，音韻照合も）が行なわれていないので，
  196                    * 還元動作を実行する必要なし．
  197                    * このようなセルは，newcelltop につないでおく．
  198                    */
  199                   if(!cp->shifted)
  200                   {
  201                       cp->next = newcelltop;
  202                       newcelltop = cp;
  203                       continue;
  204                   }
  205   #endif
  206
  207                   /*
  208                    * 現在の状態番号 state に対応する
  209                    * LR解析表の各動作項の走査．
  210                    */
  211                   for(ap = ActionTable[state]; ap != NULL; ap = ap->next)
  212                   {
  213                       /*
  214                        * LR解析表の動作は何かを調べる．
  215                        */
  216                       action = ap->action;
  217
  218                       /*
  219                        * LR解析表の現在の状態 state に還元動作以外の動作が
  220                        * あれば，other_action という変数を 0 以外にする．
  221                        */
  222                       if(IsACCEPT(action) || IsSHIFT(action))
  223                           other_action++;
  224                       /*
  225                        * 還元動作であれば，それを実行．
  226                        */
  227                       else if(IsREDUCE(action))
  228                       {
  229                           /*
  230                            * 現在の仮説セルをコピーして，新しい仮説セルを作成．
  231                            */
  232                           newcp = getcopy1(cp);
  233
  234                           /*
  235                            * instack の最上段が「単語区切り」(WORD_DELIMITER)
  236                            * でないならば，「単語区切り」を入れる．
  237                            */
  238                           if(instacktop(newcp) != WORD_DELIMITER)
  239                               inpush(WORD_DELIMITER, newcp);
  240
```

```
241                         /*
242                          * instack がオバーフロウしたら、しょうがないから
243                          * 処理を止める。
244                          */
245                         if(newcp->instackptr >= IN_STACKSIZE)
246                         {
247                             printf("Input stack overflow\n");
248                             goto END_HMMLR;
249                         }
250
251                         /*
252                          * 還元動作で用いられる文法規則の右辺の長さを
253                          * m に代入する。
254                          */
255                         m = GrammarTable[REDUCE(action)]->length;
256
257                         /*
258                          * LRの状態スタックから m 個の状態を取り除く。
259                          */
260                         npop(m, newcp);
261
262                         /*
263                          * GOTOテーブルから、次の状態を探して、
264                          * newstate に代入する。
265                          */
266                         newstate = GotoState(ActionTable, stacktop(newcp),
267                             GrammarTable[REDUCE(action)]->lhs);
268
269                         /*
270                          * newstate をLRの状態スタックにプッシュ。
271                          */
272                         push(newstate, newcp);
273
274 #if GTRACE
275                         /*
276                          * 還元動作で使われた文法規則を覚えておく。
277                          */
278                         gpush(REDUCE(action), newcp);
279 #endif
280
281 #if TWO_LEVEL
282                         /*
283                          * 2段階LRの場合、次の if 文が真であれば、
284                          * 予測された文節カテゴリに還元されたことを示す。
285                          */
286                         if(topcat[GrammarTable[REDUCE(action)]->lhs] == 1)
287                         {
288                             /*
289                              * このコメント部分は、デバッグ用コード。
290                              * printf("Recognized %s\n",
291                              * SymbolTable[GrammarTable[REDUCE(action)]->lhs]);
292                              */
293
294                             /*
295                              * 文節カテゴリ名を topcat に代入。
296                              */
297                             newcp->topcat = GrammarTable[REDUCE(action)]->lhs;
298                         }
299 #endif
300                         /*
301                          * 還元動作が連続して行なわれる可能性があるので、
302                          * 新しいセル newcp を celltop に戻しておく。
303                          */
304                         newcp->next = celltop;
305                         celltop = newcp;
306                     }
307                 }
308                 if(other_action)
309                 {
310                     /*
311                      * 次に、移動あるいは受理動作を行なうために、
312                      * セル cp は newcelltop につなげておく。
313                      */
314                     cp->next = newcelltop;
315                     newcelltop = cp;
316                 }
317                 else
318                 {
319                     /*
320                      * LR解析器の状態 state に還元動作以外の動作がなければ、
321                      * セル cp はもう不用なので、自由セル・リスト freecellp
322                      * に戻しておく。
323                      */
324                     cp->next = freecellp;
325                     freecellp = cp;
326                 }
327             }
328
329             /*
330              * celltop と newcelltop の入れ替え。
331              */
332             celltop = newcelltop;
333             newcelltop = NULL;
334
335             /*
336              * 還元動作を行なうことにより、認識仮説の数が増えるので、
337              * 枝刈りを行なう。
338              */
339             beam_search(&celltop, globalbeam, 'R');
340
341 #if BEST_FIRST
342             /*
343              * Best-first 探索の場合、最良の仮説セル top_cell
344              * とその尤度 top_prob を求める。
345              * 最初は、とりあえず、celltop の先頭のセルを最良としておき、
346              * 以下の for ループで、最良のものを見つけ出す。
347              */
348             top_cell = celltop;
349             top_prob = celltop->bestprob;
350             for(cpp = celltop; ; cpp = cpp->next)
351             {
352                 /*
353                  * 還元動作のループを既に終了したので、
354                  * 変数 shifted をリセットしておく。
355                  */
356                 cpp->shifted = 0;
357
358                 /*
359                  * セル top_cell よりも良いセル cpp が見つかったので、
360                  * top_cell, top_prob を再設定する。
```

```
361                 */
362                 if(cpp->bestprob < top_prob)
363                 {
364                         top_cell = cpp;
365                         top_prob = cpp->bestprob;
366                 }
367
368                 /*
369                  * cpp が celltop リストの最後のセルであれば、
370                  * このループを抜ける。
371                  */
372                 if(cpp->next == NULL)
373                         break;
374             }
375
376             /*
377              * 最良の仮説セルを celltop の先頭に持ってくる。
378              */
379             cpp->next = celltop;
380             celltop = top_cell;
381             while(1)
382             {
383                 if(cpp->next == top_cell)
384                 {
385                         cpp->next = NULL;
386                         break;
387                 }
388                 cpp = cpp->next;
389             }
390
391             /*
392              * 変数 acc_flag で、受理されたセルがあるかどうかを覚える。
393              * この変数を初期化。
394              */
395             acc_flag = 0;
396 #endif
397
398             /*
399              * 認識仮説の数を 0 で初期化。
400              */
401             count = 0;
402
403             /*
404              * 以下の while ループで、LR パーザの受理、移動動作を行なう。
405              */
406             while(celltop)
407             {
408                 /*
409                  * celltop の先頭のセル cp を取り出す。
410                  */
411                 cp = celltop;
412                 celltop = celltop->next;
413
414                 /*
415                  * セル cp から分岐する音韻数を数えるための変数 offs を 0 で初期化。
416                  * また、cp からの分岐を保持するためのリスト localcellp を初期化。
417                  */
418                 offs = 0;
419                 localcellp = NULL;
420
421                 /*
422                  * LR解析表の現在の状態を state に代入。
423                  */
424                 state = stacktop(cp);
425
426                 /*
427                  * 現在の状態番号 state に対応する
428                  * LR解析表の各動作項の走査。
429                  */
430                 for(ap = ActionTable[state]; ap != NULL; ap = ap->next)
431                 {
432                     /*
433                      * LR解析表の動作は何かを調べる。
434                      */
435                     action = ap->action;
436
437                     /*
438                      * 受理動作を行なう。
439                      */
440                     if(IsACCEPT(action))
441                     {
442                         /*
443                          * 現在の仮説セルをコピーして、新しい仮説セルを作成。
444                          */
445                         newcp = getcopyl(cp);
446
447                         /*
448                          * 非声の最後の母音に対する照合結果は、_start, _end 等
449                          * アンダバーで始める変数に格納されている。
450                          * これから、最後の無音の照合を行なうにあたり、これらの
451                          * 値を start, end 等アンダバーで始まらない変数に
452                          * コピーしておく。
453                          */
454                         if(newcp->_flag)
455                         {
456                             newcp->start = cp->_start;
457                             newcp->end = cp->_end;
458                             newcp->bestprob = cp->_bestprob;
459                             newcp->bestpoint = cp->_bestpoint;
460                             for(i = 0; i <= N_frames; i++)
461                                 newcp->prob[i] = cp->_prob[i];
462                         }
463
464                         /*
465                          * 入力音声の最後の無音を照合する。
466                          */
467                         verify_end(newcp, -(int_threshold));
468
469                         /*
470                          * newcp->end が入力音声長に達していれば、
471                          * この仮説を受理する。
472                          */
473                         if(newcp->end == N_frames)
474                         {
475                             /*
476                              * 受理された仮説を acceptedcellp につなぐ。
477                              */
478                             newcp->next = acceptedcellp;
479                             acceptedcellp = newcp;
480
```

LINE # SOURCE TEXT

```
481                             /*
482                              * 入力音声の最後から、end_free 個のフレームだけ
483                              * さかのぼり、その中から最もいい確率を探す。
484                              */
485                             best((newcp->end)-end_free, newcp->end,
486                               newcp->prob, &newcp->bestprob, &newcp->bestpoint);
487                             newcp->bestprob = -(newcp->bestprob);
488 #if L_MODEL
489                             /*
490                              * 統計的言語モデルを使って仮説の尤度を、
491                              * 再計算する。
492                              */
493                             if(Lmodel)
494                             {
495                                 newcp->hmmprob = newcp->bestprob;
496                                 newprob(newcp, 1);
497                             }
498 #endif
499
500 #if BEST_FIRST
501                             /*
502                              * 受理された仮説セルあり。
503                              */
504                             newcp->shifted = 0;
505                             ++acc_flag;
506 #endif
507                         }
508                         /*
509                          * newcp は音声の終端に達していないので、受理されない。
510                          * newcp はいらなくなったので、自由セル・リストに戻しておく。
511                          */
512                         else
513                         {
514                             newcp->next = freecellp;
515                             freecellp = newcp;
516                         }
517                     }
518
519                     /*
520                      * 移動動作を行なう。
521                      * ここで、音韻照合が行なわれる。
522                      */
523                     else if(IsSHIFT(action))
524                     {
525 #if TWO_LEVEL
526                         /*
527                          * 文節間LRで予測されたカテゴリに到達する可能性
528                          * がなければ、処理をスキップ。
529                          */
530                         if(!predicted(ap->cat))
531                             continue;
532 #endif
533
534                         /*
535                          * 現在の仮説セルをコピーして、新しい仮説セルを作成。
536                          */
537                         newcp = getcopy2(cp);
538
539                         /*
540                          * 発声の最後の母音の可能性があれば、_flag がセットされる。
541                          * このフラグをリセットしておく。
542                          */
543                         newcp->_flag = 0;
544
545                         /*
546                          * 音韻照合するモデルを選ぶ。
547                          */
548                         select_model(newcp, SymbolTable[ap->input],
549                                                 phone, &flg_vow);
550
551                         /*
552                          * 関数 check_end で、文の終りの可能性があるかどうかを
553                          * 調べる。
554                          */
555                         if(check_end(phone, SHIFT(action)))
556                         {
557                             /*
558                              * 文の終りの可能性があれば、語尾の母音モデル
559                              * (例えば、aj,oj) を駆動する。
560                              */
561                             ++newcp->_flag;
562                             strcpy(phone2, phone);
563                             strcat(phone2, "j");
564
565                             /*
566                              * 語尾のモデルで音韻照合を行なう。
567                              * 音韻照合の結果は、_start, _end 等
568                              * アンダーバーで始まる変数に格納する。
569                              */
570                             st = verify(phone2, flg_vow, -(int_threshold),
571                                         &newcp->_start, &newcp->_end,
572                                         &newcp->_bestprob, &newcp->_bestpoint,
573                                         newcp->_prob);
574                         }
575
576                         /*
577                          * (語尾のモデル以外の) 音韻照合を行なう。
578                          * 戻り値 st が0であれば、音韻照合失敗。
579                          */
580                         st = verify(phone, flg_vow, -(int_threshold),
581                                 &newcp->start, &newcp->end,
582                                 &newcp->bestprob, &newcp->bestpoint,
583                                 newcp->prob);
584
585                     END_VERIFY:  /* このバージョンでは未使用 */
586
587                         /*
588                          * 音韻照合した音韻を instack にプッシュし、
589                          * 記憶しておく。
590                          */
591                         inpush(ap->input, newcp);
592
593                         /*
594                          * instack がオーバーフロウしたら、しょうがないから
595                          * 処理を止める。
596                          */
597                         if(newcp->instackptr >= IN_STACKSIZE)
598                         {
599                             printf("Input stack overflow\n");
600                             goto END_HMMLR;
```

```
LINE #  SOURCE TEXT
601                         }
602
603                         /*
604                          * st が０でなければ、音韻照合は成功したことを示す。
605                          */
606                         if(st)
607                         {
608 #if L_MODEL
609                             /*
610                              * 統計的言語モデルを使って仮説の尤度を、
611                              * 再計算する。
612                              */
613                             if(Lmodel)
614                                 newprob(newcp, 0);
615 #endif
616
617                             /*
618                              * LR解析表の次の状態番号をプッシュする。
619                              */
620                             push(SHIFT(action), newcp);
621
622                             /*
623                              * 現在の仮説 newcp を localcellp につなぐ。
624                              */
625                             newcp->next = localcellp;
626                             localcellp = newcp;
627 #if BEST_FIRST
628                             /*
629                              * セル newcp に対し、音韻照合が行なわれたことを
630                              * 覚えておく。
631                              */
632                             ++newcp->shifted;
633 #endif
634                             /*
635                              * 局所的な分岐（１つのセルからの分岐）
636                              * を数えるための変数 offs をインクリメント。
637                              */
638                             ++offs;
639                         }
640                         /*
641                          * 音韻照合が失敗したら、仮説 newcp を
642                          * 自由セル・リスト freecellp に戻す。
643                          */
644                         else
645                         {
646                             newcp->next = freecellp;
647                             freecellp = newcp;
648                         }
649                     }
650                 }
651
652                 /*
653                  * 局所的ビームサーチ（１つのセルからの分岐の制限）を実行。
654                  */
655                 if(offs)
656                 {
657                     /*
658                      * offs が localbeam よりも大きければ、ビームサーチ
659                      * を行ない、枝刈りを行なう。
660                      */
661                     if(offs > localbeam)
662                         beam_search(&localcellp, localbeam, 'L');
663
664                     /*
665                      * 現在の仮説の総数をカウントし、変数 count に入れる。
666                      */
667                     for(cpp = localcellp; cpp->next != NULL; cpp = cpp->next)
668                         ++count;
669                     ++count;
670
671                     /*
672                      * localcellp 上のセルすべてを、newcelltop につなぐ。
673                      * この操作により、localcellp は空になる。
674                      */
675                     cpp->next = newcelltop;
676                     newcelltop = localcellp;
677                     localcellp = NULL;
678                 }
679
680                 /*
681                  * セル cp から分岐する仮説は、すべて処理し終ったので、
682                  * cp を自由セル・リストに戻す。
683                  */
684                 cp->next = freecellp;
685                 freecellp = cp;
686 #if BEST_FIRST
687                 /*
688                  * Best-first 探索を行なっているは、最良の仮説を処理するだけで
689                  * よいので、ループから抜ける。
690                  */
691                 break;
692 #endif
693             }
694
695 #if BEST_FIRST
696             /*
697              * Best-first 探索の場合、すべての仮説セルを
698              * celltop につなぐ。
699              */
700             if(newcelltop)
701             {
702                 for(cpp = newcelltop; cpp->next != NULL; cpp = cpp->next)
703                     ;
704                 cpp->next = celltop;
705                 celltop = newcelltop;
706             }
707             newcelltop = NULL;
708
709             /*
710              * 現在の仮説の総数をカウントする。
711              */
712             count = 0;
713             for(cpp = celltop; cpp != NULL; cpp = cpp->next)
714                 ++count;
715
716             /*
717              * もはやこれ以上、仮説が得られなければ、認識処理終了。
718              */
719             if(count == 0)
720                 break;
```

LINE # SOURCE TEXT

```
721
722             /*
723              * 一番最近の音韻照合で受理された仮説があれば、以下の処理を実行。
724              */
725             if(acc_flag)
726             {
727                 /*
728                  * 最良の仮説セルの持つ尤度を調べ、top_prob に代入。
729                  */
730                 top_prob = celltop->bestprob;
731                 for(cpp = celltop; cpp != NULL; cpp = cpp->next)
732                 {
733                         if(cpp->bestprob < top_prob)
734                                 top_prob = cpp->bestprob;
735                 }
736
737                 /*
738                  * 同じ音韻列を持った認識仮説を削除する。
739                  */
740                 rm_dupl(&acceptedcellp);
741
742                 /*
743                  * 以下のループで、受理された仮説を表示する。
744                  */
745                 for(cpp = acceptedcellp; cpp != NULL; cpp = cpp->next)
746                 {
747                         if(!cpp->shifted && cpp->bestprob <= top_prob)
748                         {
749                                 ++cpp->shifted;
750                                 cellstring(1, 1, cpp, buf);
751                                 printf("ACCEPT: ");
752                                 printf("%-25s", buf);
753                                 printf(" (prob = %8.5lf)", (double)(cpp->bestprob)*log_henkan);
754                                 printf("\n");
755
756                                 /*
757                                  * 受理された仮説の総数を示す変数 acc_count
758                                  * をインクリメント。
759                                  */
760                                 ++acc_count;
761
762                                 /*
763                                  * 受理された仮説が Nbest 個に達すれば、
764                                  * 認識処理を終了。
765                                  */
766                                 if(acc_count >= Nbest)
767                                         goto END_HMMLR;
768                         }
769                 }
770             }
771
772 #else
773             /*
774              * celltop と newcelltop を入れ替える。
775              */
776             celltop = newcelltop;
777             newcelltop = NULL;
778
779             /*
780              * もはやこれ以上、仮説が得られなければ、認識処理終了。
781              */
782             if(count == 0)
783                 break;
784 #endif
785
786             /*
787              * 無音用の threshold を、通常のものに再設定。
788              */
789             if(threshold == threshold_0)
790             {
791                 threshold = originalthreshold;
792                 int_threshold = (int)(threshold / log_henkan);
793             }
794
795             /*
796              * 認識仮説の総数が、globalbeam よりも多ければ、
797              * ビームサーチによる枝刈りを行なう。
798              */
799             if(count > globalbeam)
800             {
801                 beam_search(&celltop, globalbeam, 'G');
802
803                 /*
804                  * dynamicthreshold が設定されていれば、
805                  * threshold を再設定する。
806                  * 古いバージョンのなごり。現在、あまり意味なし。
807                  */
808                 if(dynamicthreshold)
809                 {
810                     for(cpp = celltop; cpp->next != NULL; cpp = cpp->next)
811                         ;
812                     int_tx = (cpp->bestprob + int_threshold) / 2;
813
814                     if(int_tx >= int_minimalthreshold)
815                         int_threshold = int_tx;
816                     else
817                         int_threshold = int_minimalthreshold;
818                     threshold = (double)int_threshold * log_henkan;
819                 }
820             }
821
822 #if 1
823             /*
824              * デバッグ用のコード。
825              * 認識仮説の上位候補を表示する。
826              */
827             if(Verbose)
828             {
829                 sortcell(&celltop);
830                 for(i = 0, cp = celltop; cp; cp = cp->next, i++)
831                 {
832                     if(i > 10) break;
833                     cellstring(1, 0, cp, buf);
834                     printf("[%d] %d %s prob=%lf thr=%lf state=%d (%d,%d)\n",
835                       Depth, i+1, buf, (double)cp->bestprob * log_henkan, threshold,
836                       stacktop(cp), cp->start, cp->end);
837                 }
838                 printf("-------------------------------------------------\n");
839             }
840 #endif
```

```
LINE #  SOURCE TEXT
841             }
842
843     END_HMMLR:
844
845     #if !TWO_LEVEL
846             /*
847              * 同じ音韻列を持った認識仮説を削除する．
848              */
849             rm_dupl(&acceptedcellp);
850     #endif
851
852       /*
853        * 認識された候補を表示する．
854        */
855       printf("*************************************************************\n");
856       printf("Parsing time: CPU-time = %ld msec, Elapsed-time = %ld sec.\n",
857               cputime(), realtime());
858       printf("Total-verify = %d, Depth = %d\n", total_verify, Depth);
859       ShowAcceptedParse(&acceptedcellp, Nbest);
860       printf("*************************************************************\n\n");
861       fflush(stdout);
862     }
863
864     /*
865      * 音韻名から照合するHMMのモデル名を得る．
866      */
867     select_model(cp, input, phone, flg_vow)
868     Cell    *cp;            /* 現在の仮説の入ったセル */
869     char    *input;         /* 音韻名 */
870     char    *phone;         /* 照合するHMMのモデル名 */
871     int     *flg_vow;       /* check_vow() 関数から返される値，
872                                check_vow() 関数を参照のこと */
873     {
874             strcpy(phone, input);
875
876             /*
877              * 語中の g に対しては，モデル ng を用いる．
878              * cp->instackptr が0でないということは，
879              * これまでになんらかの音韻が照合されたということを
880              * 意味している．
881              */
882             if(streq(phone, "g") && cp->instackptr != 0)
883                     strcpy(phone, "ng");
884
885             if(instacktop(cp) != WORD_DELIMITER)
886             {
887                     select_bgn_end(cp, phone);
888                     *flg_vow = check_vow(cp);
889             }
890             else
891                     *flg_vow = 0;
892     }
893
894     /*
895      * 実際に音韻照合を行なう音韻名を求める．
896      */
897     select_bgn_end(cp, phone)
898     Cell    *cp;
899     char    *phone;
900     {
901             /*
902              * 以下のコードで，cp->instackptr == 0 とは，発声の最初の
903              * 音韻であることを示している．
904              * もし，発声の最初であれば，ch, ts, p, ... 等の音韻に対しては，
905              * 語頭のモデル chl, tsl, pl, ... を用いる．
906              */
907             if(streq(phone, "ch") && cp->instackptr == 0)
908                     strcpy(phone, "chl");
909             else if(streq(phone, "ts") && cp->instackptr == 0)
910                     strcpy(phone, "tsl");
911             else if(streq(phone, "p") && cp->instackptr == 0)
912                     strcpy(phone, "pl");
913             else if(streq(phone, "t") && cp->instackptr == 0)
914                     strcpy(phone, "tl");
915             else if(streq(phone, "k") && cp->instackptr == 0)
916                     strcpy(phone, "kl");
917             else if(streq(phone, "b") && cp->instackptr == 0)
918                     strcpy(phone, "bl");
919             else if(streq(phone, "d") && cp->instackptr == 0)
920                     strcpy(phone, "dl");
921             else if(streq(phone, "r") && cp->instackptr == 0)
922                     strcpy(phone, "rl");
923             /*
924              * 語中の gy に対しては，ngy というモデルを用いる．
925              */
926             else if(streq(phone, "gy"))
927             {
928                     if(cp->instackptr == 0)
929                             strcpy(phone, "gy");
930                     else
931                             strcpy(phone, "ngy");
932             }
933     }
934
935     /*
936      * 母音に対しては，２種類の継続時間長パラメータが用意されている．
937      * 前の音韻が何であるかによって，どちらのパラメータを用いるかが決まる．
938      * 関数 check_vow では，前の音韻が何であったかを調べ，
939      * いずれのパラメータを用いるかを決める．
940      */
941     check_vow(cp)
942     Cell    *cp;
943     {
944             int     flg = 1;
945
946             if(cp->instackptr == 0)                                 flg = 0;
947             else if(streq(SymbolTable[instacktop(cp)], "a"))  flg = 0;
948             else if(streq(SymbolTable[instacktop(cp)], "i"))  flg = 0;
949             else if(streq(SymbolTable[instacktop(cp)], "u"))  flg = 0;
950             else if(streq(SymbolTable[instacktop(cp)], "e"))  flg = 0;
951             else if(streq(SymbolTable[instacktop(cp)], "o"))  flg = 0;
952             else if(streq(SymbolTable[instacktop(cp)], "N"))  flg = 0;
953             else if(streq(SymbolTable[instacktop(cp)], "Ui")) flg = 0;
954             else if(streq(SymbolTable[instacktop(cp)], "Uu")) flg = 0;
955             else if(streq(SymbolTable[instacktop(cp)], "aa")) flg = 0;
956             else if(streq(SymbolTable[instacktop(cp)], "ii")) flg = 0;
957             else if(streq(SymbolTable[instacktop(cp)], "uu")) flg = 0;
958             else if(streq(SymbolTable[instacktop(cp)], "ee")) flg = 0;
959             else if(streq(SymbolTable[instacktop(cp)], "oo")) flg = 0;
960             else if(streq(SymbolTable[instacktop(cp)], "ei")) flg = 0;
```

LINE # SOURCE TEXT

```
961         else if(streq(SymbolTable[instacktop(cp)], "ou")) flg = 0;
962         else if(streq(SymbolTable[instacktop(cp)], "s"))  flg = 0;
963         else if(streq(SymbolTable[instacktop(cp)], "sh")) flg = 0;
964         else if(streq(SymbolTable[instacktop(cp)], "h"))  flg = 0;
965         else if(streq(SymbolTable[instacktop(cp)], "ch")) flg = 0;
966         else if(streq(SymbolTable[instacktop(cp)], "ts")) flg = 0;
967         else if(streq(SymbolTable[instacktop(cp)], "p"))  flg = 0;
968         else if(streq(SymbolTable[instacktop(cp)], "k"))  flg = 0;
969         return(flg);
970 }
971
972 /*
973  * 文の終りの可能性をチェックする．
974  * これから照合しようとする音韻 phone が文の終りに来る
975  * 可能性があれば 1 を返す．
976  */
977 check_end(phone, state)
978 char    *phone;         /* 音韻名 */
979 int     state;          /* LR解析表の状態 */
980 {
981         int     newstate, ok = 0;
982         register Action *ap;
983
984         /*
985          * 日本語では，文の終りは子音ではないので，
986          * これから照合する音韻 phone が母音かどうかをチェックする．
987          */
988         if(streq(phone, "a")  || streq(phone, "i")  || streq(phone, "Ui")
989         || streq(phone, "u")  || streq(phone, "Uu") || streq(phone, "e")
990         || streq(phone, "o")  || streq(phone, "N")  || streq(phone, "aa")
991         || streq(phone, "ii") || streq(phone, "uu") || streq(phone, "ee")
992         || streq(phone, "oo") || streq(phone, "oi") || streq(phone, "ou"))
993                 ++ok;
994
995         /*
996          * 母音でなければ，即リターン．
997          */
998         if(!ok)
999                 return(0);
1000
1001        /*
1002         * LR解析表の動作欄を調べ，現在の状態 state の欄に
1003         * ENDSYMBOL 記号 ($) で還元動作の指定されているものが
1004         * あるかどうかをチェックする．
1005         */
1006        for(ap = ActionTable[state]; ap != NULL; ap = ap->next)
1007        {
1008                if(IsREDUCE(ap->action)
1009                && streq(SymbolTable[ap->input], ENDSYMBOL))
1010                        return(1);
1011        }
1012        return(0);
1013 }
1014
1015 /*
1016  * セル cp の持つ音韻列仮説を buf に代入．
1017  * delimit が 0 でなければ，単語境界にハイフンをそ挿入する．
1018  * acc が 0 でなければ，仮説の最後の単語境界記号は無視．
1019  */
1020 cellstring(delimit, acc, cp, buf)
1021 int     delimit;
1022 int     acc;
1023 Cell    *cp;
1024 char    *buf;
1025 {
1026         register int    i;
1027
1028         buf[0] = 0x00;
1029         if(cp->instackptr == 0)
1030                 strcat(buf, "[Empty]");
1031         else
1032         {
1033                 for(i = 0; i < cp->instackptr; i++)
1034                 {
1035                         if(acc && i + 1 == cp->instackptr)
1036                                 break;
1037                         if(cp->instack[i] == WORD_DELIMITER)
1038                         {
1039                                 if(delimit)
1040                                         strcat(buf, "-");
1041                         }
1042                         else
1043                         {
1044                             if(streq(SymbolTable[cp->instack[i]], "Ui"))
1045                                 strcat(buf, "i");
1046                             else if(streq(SymbolTable[cp->instack[i]], "Uu"))
1047                                 strcat(buf, "u");
1048                             else
1049                                 strcat(buf, SymbolTable[cp->instack[i]]);
1050                         }
1051                 }
1052         }
1053 }
1054
1055 /*
1056  * セルのリスト top から尤度の高いセル max 個を表示する．
1057  * HMM-LR の認識候補を表示するのに使用．
1058  */
1059 ShowAcceptedParse(top, max)
1060 Cell    **top;
1061 int     max;
1062 {
1063         register int    i, j;
1064         register Cell   *cp;
1065         char    buf[IN_STACKSIZE*2];
1066
1067         /*
1068          * セルのリストを尤度順にソートする．
1069          */
1070         sortcell(top);
1071
1072         /*
1073          * 認識順位，音韻列，尤度の表示．
1074          */
1075         for(cp = *top, i = 0; i < globalbeam && cp; cp = cp->next)
1076         {
1077 #if TWO_LEVEL
1078                 if(cp->topcat == -1)
1079                         continue;
1080 #endif
```

LINE # | SOURCE TEXT

```
1081
1082                    if(i >= max)
1083                            break;
1084
1085                    /* 認識順位 */
1086                    printf("%3d: ", i + 1);
1087
1088                    /* 音韻列 */
1089                    callstring(1, 1, cp, buf);
1090                    printf("%-25s", buf);
1091
1092                    /* 尤度 */
1093                    printf(" (prob = %8.5lf)", (double)(cp->bestprob)*log_henkan);
1094
1095    #if TWO_LEVEL
1096                    /*
1097                     * 2段階LRの場合，文節カテゴリ名を表示．
1098                     */
1099                    printf(" %s", SymbolTable[cp->topcat]);
1100    #endif
1101
1102                    printf("\n");
1103                    fflush(stdout);
1104                    ++i;
1105            }
1106    }
1107
1108    /*
1109     * 尤度配列 prob の start と end で指定された区間から
1110     * 正規化尤度の最も大きな値および位置を探し bestprob, bestpoint に代入する．
1111     */
1112    best(start, end, prob, bestprob, bestpoint)
1113    int     start, end, *prob, *bestprob, *bestpoint;
1114    {
1115            int     i, px;
1116            double  rint();
1117
1118            *bestprob = (int)rint((double)prob[start] / (double)start);
1119            *bestpoint = start;
1120            for(i = start+1; i <= end; i++)
1121            {
1122                    px = (int)rint((double)prob[i] / (double)i);
1123                    if(px > *bestprob)
1124                    {
1125                            *bestprob = px;
1126                            *bestpoint = i;
1127                    }
1128            }
1129    }
```

LINE # SOURCE TEXT

```
/*------------------------------------------------------------------
 *
 *      セルの操作関数
 *
 *       ATR Interpreting Telephony Research Laboratories.
 *
 *      History
 *       24-Jun-1988, Created by Kenji Kita.
 *        6-Jul-1988, Heavily modified.
 *       24-Dec-1988, Fixed getcopy() by Hanazawa.
 *       29-Jun-1991, for Two-level LR.
 *
 *------------------------------------------------------------------*/

#define CELL_MAIN

#include <stdio.h>
#include "config.h"
#include "HMM.h"
#include "cell.h"

/*
 * セルの領域（totalcells 個のセル）を確保する．
 */
init_cell()
{
        char    *calloc( );

        if((cellstart = (Cell*)calloc(totalcells, sizeof(Cell))) == NULL)
                fatal_error("No memory for cells.\n");
}

/*
 * セル領域をHMM-LRが使えるように，設定し直す．
 * 認識の最初に，この関数を呼び出す必要あり．
 */
reinitcell()
{
        register int    i;
        register Cell   *p;
        Cell    *getcell();

        /*
         * セルをつないで，大きなリスト（自由セル・リスト）を作る．
         * セルが必要になることに，自由セル・リストからセルを１つづつ
         * 取り出してくる．
         */
        cellend = cellstart + totalcells;
        for(p = cellstart; p < cellend - 1; p++)
                p->next = p + 1;
        p->next = NULL;  /* 自由セルの終り */

        /*
         * 各リストの初期化．
         */
        freecellp = cellstart;
        newcelltop = NULL;
        acceptedcellp = NULL;

        /*
         * セルを１つ取ってくる．
         */
        celltop = getcell();

        /*
         * 音声の始終端を０に設定．
         */
        celltop->start = 0;
        celltop->end = 0;

        celltop->next = NULL;

        /*
         * LRの状態スタックを初期化．
         */
        clrstack(celltop);

        /*
         * 認識音韻列スタックを初期化．
         */
        clrinstack(celltop);

#if GTRACE
        /*
         * 文法履歴スタックの初期化．
         */
        clrgstack(celltop);
#endif

#if SYLLABLE_NGRAM
        /*
         * 音節trigram計算用変数の初期化．
         */
        clrmstack(celltop);
        celltop->symbol[0] = 0x00;
        celltop->moraprob = 0.0;
#endif

        /*
         * HMM尤度配列の初期化．
         */
        celltop->prob[0] = 0;
        for(i = 1; i <= N_frames; i++)
                celltop->prob[i] = MIN_INT;

        /*
         * LRの状態スタックに初期状態０をプッシュ．
         */
        push(0, celltop);

#if TWO_LEVEL
        celltop->topcat = -1;
#endif
}

/*
 * 自由セル・リストからセルを１つ取ってくる．
 */
Cell    *getcell()
```

LINE # SOURCE TEXT

```
121  {
122          Cell    *p;
123
124          if(freecellp == NULL)
125                  GC();
126          p = freecellp;
127          freecellp = freecellp->next;
128          return(p);
129  }
130
131  /*
132   * セル p のコピーを作る．
133   */
134  Cell    *getcopy1(p)
135  Cell    *p;
136  {
137          register int    i;
138          Cell    *q;
139
140          if(freecellp == NULL)
141                  GC();
142          /*
143           * 自由セル・リストから1個セルを取ってくる．
144           */
145          q = freecellp;
146          freecellp = freecellp->next;
147          /*
148           * 取ってきたセル q に p の内容をコピー．
149           */
150          bcopy((char *)p, (char *)q, sizeof(Cell));
151          return(q);
152  }
153
154  /*
155   * セル p のコピーを作る．
156   * 最初の部分は getcopy1 と同じだが，最後の部分が違うので注意．
157   */
158  Cell    *getcopy2(p)
159  Cell    *p;
160  {
161          register int    i;
162          Cell    *q;
163
164          if(freecellp == NULL)
165                  GC();
166          /*
167           * 自由セル・リストから1個セルを取ってくる．
168           */
169          q = freecellp;
170          freecellp = freecellp->next;
171          /*
172           * 取ってきたセル q に p の内容をコピー．
173           */
174          bcopy((char *)p, (char *)q, sizeof(Cell));
175
176          q->_start = p->start;
177          q->_end = p->end;
178          q->_bestprob = p->bestprob;
179          q->_bestpoint = p->bestpoint;
180          for(i = 0; i <= N_frames; i++)
181                  q->_prob[i] = p->prob[i];
182          return(q);
183  }
184
185  /*
186   * 自由セル・リストが空になったときに，
187   * 尤度の低いセルを回収してきて，再利用できるようにする．
188   */
189  GC()
190  {
191          extern int      Nbest;
192          int     beam;
193
194          beam = globalbeam;
195    start:
196          beam_search(&newcelltop, beam, 'G');
197          if(!freecellp)
198          {
199                  beam_search(&acceptedcellp, Nbest, 'G');
200                  if(freecellp == NULL)
201                  {
202                          --beam;
203                          if(!beam)
204                                  fatal_error("Cell exhausted.\n");
205                          goto start;
206                  }
207          }
208  }
209
210  /*
211   * 重複した音韻列仮説を持つ候補を削除する．
212   */
213  rm_dupl(top)
214  Cell    **top;
215  {
216          register int    i;
217          register Cell   *p, *q, *r, *cp;
218          int     ok;
219
220          for(cp = NULL, p = *top; p != NULL; )
221          {
222                  r = p;
223                  p = p->next;
224                  ok = 1;
225                  for(q = cp; q != NULL; q = q->next)
226                  {
227                          if(r->instackptr != q->instackptr)
228                                  continue;
229                          for(i = 0; i < r->instackptr; i++)
230                          {
231                                  if(r->instack[i] != q->instack[i])
232                                          break;
233                          }
234                          if(i == r->instackptr)
235                          {
236                                  ok = 0;
237                                  break;
238                          }
239                  }
240                  if(!ok)
```

```
LINE #  SOURCE TEXT
241                 {
242                     q->bestprob =
243                       (r->bestprob < q->bestprob ? r->bestprob : q->bestprob);
244
245                         r->next = freecellp;
246                         freecellp = r;
247                 }
248                 else
249                 {
250                         r->next = cp;
251                         cp = r;
252                 }
253         }
254         *top = cp;
255 }
256
257 /*
258   ・セルのリストを光度順にソートする
259   ・セルの bestprob の小さい順に並べる
260   */
261 sortcell(cell)
262 Cell    **cell;
263 {
264 #define insert(cp, first, last)                         \
265                         { if(!first) first = cp;        \
266                           else last->next = cp;         \
267                           last = cp; }
268
269         Cell    *cp = *cell, *lowf, *lowl, *midf, *midl, *highf, *highl;
270
271         if(cp == NULL)
272                 return;
273         lowf = midf = highf = NULL;
274         insert(cp, midf, midl);
275         for(cp = cp->next; cp != NULL; cp = cp->next)
276         {
277                 if(cp->bestprob < midf->bestprob)
278                         insert(cp, lowf, lowl)
279                 else if(cp->bestprob == midf->bestprob)
280                         insert(cp, midf, midl)
281                 else
282                         insert(cp, highf, highl)
283         }
284         if(lowf != NULL)
285         {
286                 lowl->next = NULL;
287                 sortcell(&lowf);
288                 for(cp = lowf; cp->next != NULL; cp = cp->next)
289                         ;
290                 cp->next = midf;
291                 cp = lowf;
292         }
293         else
294                 cp = midf;
295         if(highf != NULL)
296                 highl->next = NULL;
297         sortcell(&highf);
298         midl->next = highf;
299         *cell = cp;
300 }
```

LINE # SOURCE TEXT

```
/*---------------------------------------------------------------*
 *
 *      文法、ＬＲ解析表のロード等
 *
 *          ATR Interpreting Telephony Research Laboratories.
 *
 *      History
 *
 *          23-Jun-1988 Created by Kenji Kita.
 *
 *---------------------------------------------------------------*/

#define LR_MAIN

#include <stdio.h>
#include "config.h"
#include "LR.h"

/*
 * streq(s,t) : 文字列 s と t が等しいかを判断する．
 */
#define streq(s,t)      (*(s) == *(t) && strcmp(s,t) == 0)

/*
 * GOTOの行き先の状態番号を求める．
 */
GotoState(a_table, state, lhs)
Action  **a_table;      /* LR解析表 */
int     state,          /* 状態 */
        lhs;            /* 文法記号 */
{
        register Action *p;
        register int    *i, n;

        for(p = a_table[state]; p != NULL; p = p->next)
        {
                if(p->input == lhs)
                        return(GOTO(p->action));
        }
        fatal_error("Cannot find goto state. (state=%d, LHS=%s)\n",
          state, SymbolTable[lhs]);
}

/*
 * 文法規則のロード．
 */
InstallGrammar(g_table, index, left, right, prob)
Rule    **g_table;      /* 文法規則のテーブル */
char    *left, *right;  /* 左辺および右辺の文字列 */
double  prob;           /* 確率 */
{
        register int    i, *n;
        register char   *p, *q;
        char    work[64], *malloc();
        Rule    *rp;

        if(index > MAXGRAMMARS - 1)
                fatal_error("Grammar too large. Modify MAXGRAMMARS.\n");

        for(i = 0, p = right; *p != NULL; p++)
        {
                if(*p == ' ')
                        ++i;
        }
        if((rp = (Rule *)malloc(sizeof(Rule))) == NULL)
                fatal_error("No memory for cfg rule.\n");
        if((n = rp->rhs = (int *)malloc((i+2)*sizeof(int))) == NULL)
                fatal_error("No memory for cfg rule.\n");
#if STOCHASTIC_GRAMMAR
        rp->gprob = prob;
#endif
        rp->length = i + 1;
        rp->lhs = InstallSymbol(left);
        p = right;
        q = work;
        while(1)
        {
                if(*p == ' ' || *p == NULL)
                {
                        *q = NULL;
                        *n++ = InstallSymbol(work);
                        q = work;
                }
                else
                        *q++ = *p;
                if(*p == NULL)
                        break;
                ++p;
        }
        *n = -1;
        g_table[index] = rp;
}

/*
 * LR解析表のロード．
 */
#if STOCHASTIC_LR
  InstallAction(ReducePacking, a_table, state, input, action, prob)
  int           ReducePacking;  /* 還元動作をパッキングして持つか？ */
  Action        **a_table;      /* LR解析表 */
  int           state;          /* 状態 */
  char          *input;         /* 文法記号 */
  int           action;         /* 動作 */
  double        prob;           /* 確率 */
#else
  InstallAction(ReducePacking, a_table, state, input, action)
  int           ReducePacking;
  Action        **a_table;
  int           state;
  char          *input;
  int           action;
#endif
{
        char    *malloc();
        Action  *p, *q;

        if(state > MAXSTATES - 1)
                fatal_error("LR table too large. Modify MAXSTATES.\n");

        if(IsREDUCE(action) && ReducePacking)
```

```
LINE #  SOURCE TEXT
121             {
122                     for(p = a_table[state]; p != NULL; p = p->next)
123                     {
124                             if(p->action == action)
125                             {
126                                     if(streq(input, ENDSYMBOL))
127                                             p->input = HashSymbol(ENDSYMBOL);
128                                     return;
129                             }
130                     }
131                     if(!streq(input, ENDSYMBOL))
132                             input = "*";
133             }
134
135             if((p = (Action *)malloc(sizeof(Action))) == NULL)
136                     fatal_error("No memory for action.\n");
137             p->input = InstallSymbol(input);
138             p->action = action;
139             p->next = NULL;
140     #if STOCHASTIC_LR
141             p->prob = prob;
142     #endif
143             if(a_table[state] == NULL)
144                     a_table[state] = p;
145             else
146             {
147                     for(q = a_table[state]; q->next != NULL; q = q->next)
148                             ;
149                     q->next = p;
150             }
151     }
152
153     /*
154       * 記号 s を SymbolTable に格納する．
155       */
156     InstallSymbol(s)
157     char    *s;
158     {
159             int     index;
160             char    *p, *malloc();
161
162             if(SymbolTable[index = HashSymbol(s)] != NULL)
163                     return(index);
164             if((p = (char *)malloc(strlen(s)+1)) == NULL)
165                     fatal_error("No memory for symbol '%s'.\n", s);
166             strcpy(p, s);
167             SymbolTable[index] = p;
168             return(index);
169     }
170
171     /*
172       * 記号 s を格納すべき場所（SymbolTable のインデックス）を求める．
173       * 既に 記号 s が格納されていれば 記号 s のインデックスを返す．
174       */
175     HashSymbol(s)
176     char    *s;
177     {
178             char    *p;
179             int     old_index, index, n;
180
181             for(index = 0, p = s; *p != NULL; )
182                     index += *p++;
183             old_index = index = index % MAXSYMBOLS;
184             for(n = 0; SymbolTable[index] != NULL; ++n, index = (old_index+n*n) % MAXSYMBOLS)
185             {
186                     if(n != 0 && index == old_index)
187                             fatal_error("SymbolTable full.\n");
188                     if(strcmp(SymbolTable[index], s) == 0)
189                             break;
190             }
191             return(index);
192     }
```

LINE # SOURCE TEXT

```
/*---------------------------------------------------------------------*
 *
 *      文法規則を読み込む
 *
 *          ATR Interpreting Telephony Research Laboratories.
 *
 *      History
 *
 *          15-Aug-1988, Created by Kenji Kita.
 *           7-Mar-1990, Modified.
 *
 *---------------------------------------------------------------------*/

#include <stdio.h>
#include <ctype.h>
#include "config.h"
#include "LR.h"

/*
 * 文法規則は、以下の形式で書かれている。
 *
 * (左辺 (--> (右辺1 右辺2 ...)) (適用確率))
 */

#define LPAREN          '('
#define RPAREN          ')'
#define LBRACE          '{'
#define RBRACE          '}'

/*
 * 文法規則中の書き換え記号。
 */
#define REWRITESYM      "<--"

/*
 * エラー・メッセージ。
 */
#define EOFERROR        "Unexpected EOF.\n"

/*
 * streq(s,t)   文字列 s と t が等しいかを判断する。
 */
#define streq(s,t)      (*(s)==*(t) && strcmp(s,t)==0)

/*
 * 文法規則をファイル・ポインタ fp から読み込み、
 * 文法テーブル g_table に格納する。
 */
ReadRules(fp, g_table)
FILE    *fp;
Rule    **g_table;
{
        int     c, nrules = 0;
        char    lhs[BUFSIZ], rhs[BUFSIZ], buf[BUFSIZ];
        double  gprob;

        /*
         * 1回のループで、規則を1つ読み込む。
         */
        while(1)
        {
                rhs[0] = NULL;
                if(SkipTo(fp, LPAREN) == EOF)
                        break;
                /*
                 * 左辺の読み込み。
                 */
                GetToken(fp, lhs);
                /*
                 * 書き換え記号の読み込み。
                 */
                GetToken(fp, buf);
                if(!streq(buf, REWRITESYM))
                        lex_error("Missing or illegal rewriting symbol.\n");
                /*
                 * 右辺の読み込み。
                 */
                if(PeekChar(fp) != LPAREN)
                        lex_error("Missing left(open) parenthesis at RHS.\n");
                if(SkipTo(fp, LPAREN) == EOF)
                        lex_error(EOFERROR);
                while(1)
                {
                        GetToken(fp, buf);
                        strcat(rhs, buf);
                        if(PeekChar(fp) == RPAREN)
                                break;
                        strcat(rhs, " ");
                }
                if(SkipTo(fp, RPAREN) == EOF)
                        lex_error(EOFERROR);
                if(PeekChar(fp) != RPAREN)
                        lex_error("Missing rule-close parenthesis.\n");
                if(SkipTo(fp, RPAREN) == EOF)
                        lex_error(EOFERROR);
#if STOCHASTIC_GRAMMAR
                if((c = PeekChar(fp)) == EOF || c != LBRACE)
                {
                        gprob = 0.0;
                }
                else
                {
                        SkipTo(fp, LBRACE);
                        fscanf(fp, "%lf", &gprob);
                        if(PeekChar(fp) != RBRACE)
                                lex_error("Missing right brace.\n");
                        SkipTo(fp, RBRACE);
                }
#endif
                InstallGrammar(g_table, nrules, lhs, rhs, gprob);

#if STOCHASTIC_GRAMMAR
                InstallVN(lhs, nrules);
#endif

                ++nrules;
        }
#if STOCHASTIC_GRAMMAR
        init_prob();
#endif
```

LINE # | SOURCE TEXT

```
121         return(nrules);
122 }
```

LINE # SOURCE TEXT

```
/*----------------------------------------------------------------*
 *
 *      LR解析表を読み込む
 *
 *          ATR Interpreting Telephony Research Laboratories.
 *
 *      History
 *
 *          23-Jun-1988, Created by Kenji Kita.
 *          16-Feb-1990, Heavily modified.
 *
 *----------------------------------------------------------------*/

#include <stdio.h>
#include <ctype.h>
#include <math.h>
#include <sys/types.h>
#include <sys/file.h>
#include <sys/stat.h>
#include "config.h"
#include "LR.h"

#define LPAREN          '('
#define RPAREN          ')'

/*
 * streq(s,t)   文字列 s と t が等しいかを判断する．
 */
#define streq(s,t)      (*(s)==*(t) && strcmp(s,t)==0)

/*
 * 指定された文法名 GrammarName に対し，適当な文法ファイルおよびLR解析表ファイルを
 * オープンし，文法規則およびLR解析表を，それぞれ g_table, a_table に読み込む．
 * 文法ファイルは，GrammarName にファイル拡張子 .gra を付けたものであり，
 * LR解析表ファイルは，GrammarName にファイル拡張子 .tab を付けたものである．
 * 引数 ReducePacking が０でなければ，同じ状態番号にある同一規則による還元動作は
 * １個にまとめられる．
 */
ReadLR(ReducePacking, GrammarName, g_table, a_table)
int     ReducePacking;
char    *GrammarName;
Rule    **g_table;
Action  **a_table;
{
        extern int      lineno; /* ファイルの行番号 */
        int     nstates,        /* LR解析表の状態数 */
                nrules;         /* 文法規則数 */
        char    Grammarfile[80],/* 文法ファイル名 */
                LRfile[80];     /* LR解析表ファイル名 */
        FILE    *fp;

        /*
         * 文法ファイル名，LR解析表ファイル名の設定．
         */
        sprintf(Grammarfile, "%s.gra", GrammarName);
        sprintf(LRfile,      "%s.tab", GrammarName);

        /*
         * LR解析表ファイルは，文法ファイルから作られるため，
         * LR解析表ファイルの作成された日時は，文法ファイルから作られた
         * 日時に比べ，時間的に前である必要がある．
         * 以上をチェックする．
         */
        if(checktime(Grammarfile, LRfile) == 0)
                fatal_error("Grammar is old.\n");

        /*
         * 関数 ReadRules を呼んで，文法規則を読み込む．
         */
        lineno = 1;
        if((fp = fopen(Grammarfile, "r")) == NULL)
                fatal_error("Grammar file(%s) cannot open\n", Grammarfile);
        fprintf(stderr, "-> Loading grammar rules from '%s'...", Grammarfile);
        nrules = ReadRules(fp, g_table);
        fclose(fp);
        fprintf(stderr, "[%d rules]\n", nrules);

        /*
         * 関数 ReadAction を呼んで，LR解析表を読み込む．
         */
        lineno = 1;
        if((fp = fopen(LRfile, "r")) == NULL)
                fatal_error("LR file(%s) cannot open\n", LRfile);
        fprintf(stderr, "-> Loading LR parsing table from '%s'...", LRfile);
        nstates = ReadAction(ReducePacking, fp, a_table);
        fclose(fp);
        fprintf(stderr, "[%d states]\n", nstates);
}

/*
 * LR解析表を読み込んで，a_table に格納する．
 * 引数 ReducePacking については，関数 ReadLR を参照．
 */
static  ReadAction(ReducePacking, fp, a_table)
int     ReducePacking;
FILE    *fp;
Action  **a_table;
{
#define ACTMINPROB      1.e-8
        int     state, action, nstates = 0;
        char    input[64], buf[64];
#if STOCHASTIC_LR
        double  prob;
#endif

        /*
         * LR解析表の先頭（コメント部分を除く）には，
         * (slr-table) という表示があることを仮定している．
         * もし，こういう表示がなければ，このファイルは
         * LR解析表ではないと判断し，処理を止める．
         */
        SkipTo(fp, LPAREN);
        GetToken(fp, input);
        if(!streq(input, "slr-table"))
                fatal_error("Bad LR parsing table.\n");
        SkipTo(fp, RPAREN);

        /*
         * １回のループで，１つの状態に対する動作項を読み込む．
         */
```

```
LINE #  SOURCE TEXT
121             while(1)
122             {
123                     SkipTo(fp, LPAREN);
124
125                     fscanf(fp, "%d", &state);
126                     while(1)
127                     {
128                             SkipTo(fp, LPAREN);
129
130                             GetToken(fp, input);
131                             ChangeInput(input);
132
133                             GetToken(fp, buf);
134                             switch(buf[0])
135                             {
136                             case 'a': action = ACCEPT; break;
137                             case 's': action = atoi(&buf[1]); break;
138                             case 'r': action = atoi(&buf[1]) + 2*MAXSTATES; break;
139                             case 'g': action = atoi(&buf[1]) + MAXSTATES; break;
140                             default: fatal_error("Unknown action: %s\n", buf);
141                             }
142
143     #if STOCHASTIC_LR
144                             if(buf[0] == 'a' || buf[0] == 's' || buf[0] == 'r')
145                                     fscanf(fp, "%lf", prob);
146                             if(prob < ACTMINPROB)
147                                     prob = ACTMINPROB;
148                             prob = -log10(prob);
149     #endif
150
151                             SkipTo(fp, RPAREN);
152
153     #if STOCHASTIC_LR
154             InstallAction(ReducePacking, a_table, state, input, action, prob);
155     #else
156             InstallAction(ReducePacking, a_table, state, input, action);
157     #endif
158
159                             if(PeekChar(fp) == RPAREN)
160                             {
161                                     SkipTo(fp, RPAREN);
162                                     break;
163                             }
164                     }
165                     ++nstates;
166                     if(PeekChar(fp) == EOF)
167                             break;
168             }
169             return(nstates);
170     }
171
172     /*
173      * 文法および LR解析表中のいくつかのシンボルを、対応する音韻名に変換する。
174      * このコードは一見、美しくないように見えるかもしれない。
175      * しかし、これは HMM-LR の歴史的事情により、しかたなくこうしたのである。
176      */
177     static  ChangeInput(input)
178     char    *input;
179     {
180             register char   *p;
181             char    work[64];
182
183             if(streq(input, "="))
184                     strcpy(input, "N");
185             else if(streq(input, "i2"))
186                     strcpy(input, "Ui");
187             else if(streq(input, "u2"))
188                     strcpy(input, "Uu");
189             else if(streq(input, "q"))
190                     strcpy(input, "Q");
191             else if(streq(input, "q1"))
192                     strcpy(input, "Q1");
193             else if(streq(input, "q2"))
194                     strcpy(input, "Q2");
195     }
196
197     /*
198      * 2つのファイル file1 と file2 の作成された日時を比べる。
199      * file1 が file2 より、時間的に前に作成されていれば 1 を返す。
200      * そうでなければ 0 を返す。
201      */
202     static  checktime(file1, file2)
203     char    *file1, *file2;
204     {
205             struct stat     stbuf1, stbuf2;
206
207             if(stat(file1, &stbuf1) == -1)
208                     fatal_error("Can't access %s\n", file1);
209             if(stat(file2, &stbuf2) == -1)
210                     fatal_error("Can't access %s\n", file2);
211             if(stbuf1.st_mtime < stbuf2.st_mtime)
212                     return(1);
213             return(0);
214     }
```

LINE # SOURCE TEXT

```
/*-----------------------------------------------------------------------
 *
 *      文節内ＬＲ解析表を読み込む
 *
 *      本プログラムは、２段階ＬＲでのみ用いられる．
 *      基本的には、ファイル ReadLR.c と同様な処理を行なう．
 *      ReadLR.c との違いは、名動作順に対して到達可能な
 *      文節カテゴリのリストを読み込む点である．
 *
 *      ATR Interpreting Telephony Research Laboratories.
 *
 *      History
 *
 *      11-Apr-1991, Created by Kenji Kita.
 *
 *-----------------------------------------------------------------------*/

#include <stdio.h>
#include <ctypo.h>
#include <math.h>
#include <sys/types.h>
#include <sys/file.h>
#include <sys/stat.h>
#include "config.h"
#include "LR.h"

#if TWO_LEVEL

#define FALSE           0
#define TRUE            1
#define LPAREN          '('
#define RPAREN          ')'
#define LBRA            '['
#define RBRA            ']'
#define streq(s,t)      (*(s)==*(t) && strcmp(s,t)==0)

#if TWO_LEVEL
/*
 * すべての文節カテゴリを覚えておくための変数．
 * SymbolTable のインデックスが入る．
 */
int     *allcat;
#endif

ReadLR2(ReduceMethod, GrammarName, g_table, a_table)
int     ReduceMethod;
char    *GrammarName;
Rule    **g_table;
Action  **a_table;
{
        extern int      lineno;
        int     nstates, nrules, ncats;
        char    Grammarfile[80], LRfile[80];
#if TWO_LEVEL
        char    Catfile[80];
#endif
        FILE    *fp;

        sprintf(Grammarfile, "%s.gra", GrammarName);
        sprintf(LRfile,      "%s.tab", GrammarName);
#if TWO_LEVEL
        sprintf(Catfile,     "%s.cat", GrammarName);
#endif

        if(checktime(Grammarfile, LRfile) == FALSE)
                fatal_error("Grammar is old.\n");

        /*********************
         * Read grammar rules.
         *********************/
        lineno = 1;
        if((fp = fopen(Grammarfile, "r")) == NULL)
                fatal_error("Grammar file(%s) cannot open\n", Grammarfile);
        fprintf(stderr, "-> Loading grammar rules '%s'...", Grammarfile);
        nrules = ReadRules(fp, g_table);
        fclose(fp);
        fprintf(stderr, "[%d rules]\n", nrules);

#if TWO_LEVEL
        if((fp = fopen(Catfile, "r")) == NULL)
                fatal_error("Cat file(%s) cannot open\n", Catfile);
        ReadCat(fp, &allcat);
        fclose(fp);
#endif

        /*****************
         * Read LR table.
         *****************/
        lineno = 1;
        if((fp = fopen(LRfile, "r")) == NULL)
                fatal_error("LR file(%s) cannot open\n", LRfile);
        fprintf(stderr, "-> Loading LR parsing table '%s'...", LRfile);
        nstates = ReadAction2(ReduceMethod, fp, a_table);
        fclose(fp);
        fprintf(stderr, "[%d states]\n", nstates);
}

ReadAction2(ReduceMethod, fp, a_table)
int     ReduceMethod;
FILE    *fp;
Action  **a_table;
{
#define ACTMINPROB      1.e-8
        int     state, action, nstates = 0;
        char    input[64], buf[64];
#if STOCHASTIC_LR
        double  prob;
#endif
#if TWO_LEVEL
        int     cat[MAXGRAMMARS];
#endif

        SkipTo(fp, LPAREN);
        GetToken(fp, input);
        if(!streq(input, "slr-table"))
                fatal_error("Bad format in LR parsing table.\n");
        SkipTo(fp, RPAREN);

        while(1)
        {
```

LINE # SOURCE TEXT

```
121                     SkipTo(fp, LPAREN);
122
123                     fscanf(fp, "%d", &state);
124                     while(1)
125                     {
126                             SkipTo(fp, LPAREN);
127
128                             GetToken(fp, input);
129                             ChangeInput(input);
130
131                             GetToken(fp, buf);
132                             switch(buf[0])
133                             {
134                             case 'a': action = ACCEPT; break;
135                             case 's': action = atoi(&buf[1]); break;
136                             case 'r': action = atoi(&buf[1]) + 2*MAXSTATES; break;
137                             case 'g': action = atoi(&buf[1]) + MAXSTATES; break;
138                             default: fatal_error("Unknown action: %s\n", buf);
139                             }
140
141 #if STOCHASTIC_LR
142                             if(buf[0] == 'a' || buf[0] == 's' || buf[0] == 'r')
143                                     fscanf(fp, "%lf", prob);
144                             if(prob < ACTMINPROB)
145                                     prob = ACTMINPROB;
146                             prob = -log10(prob);
147 #endif
148
149 #if TWO_LEVEL
150                             if(buf[0] == 's')
151                             {
152                                     if(streq(input, "Q1") || streq(input, "Q2"))
153                                     {
154                                             register int    i;
155
156                                             for(i = 0; allcat[i] != -1; i++)
157                                                     cat[i] = allcat[i];
158                                             cat[i] = -1;
159                                     }
160                                     else
161                                     {
162                                             if(!ReadCat2(fp, cat))
163                                     fatal_error("No cat. state=%d\n", state);
164                                     }
165                             }
166 #endif
167
168                             SkipTo(fp, RPAREN);
169
170 #if STOCHASTIC_LR
171             InstallAction2(ReducePacking, a_table, state, input, action, prob);
172 #else
173             InstallAction2(ReducePacking, a_table, state, input, action, cat);
174 #endif
175
176                             if(PeekChar(fp) == RPAREN)
177                             {
178                                     SkipTo(fp, RPAREN);
179                                     break;
180                             }
181                     }
182                     ++nstates;
183                     if(PeekChar(fp) == EOF)
184                             break;
185             }
186             return(nstates);
187 }
188
189 static  ChangeInput(input)
190 char    *input;
191 {
192             register char   *p;
193             char    work[64];
194
195             if(streq(input, "="))
196                     strcpy(input, "N");
197             else if(streq(input, "i2"))
198                     strcpy(input, "Ui");
199             else if(streq(input, "u2"))
200                     strcpy(input, "Uu");
201             else if(streq(input, "q"))
202                     strcpy(input, "Q");
203             else if(streq(input, "q1"))
204                     strcpy(input, "Q1");
205             else if(streq(input, "q2"))
206                     strcpy(input, "Q2");
207 }
208
209 static  checktime(file1, file2)
210 char    *file1, *file2;
211 {
212             struct stat     stbuf1, stbuf2;
213
214             if(stat(file1, &stbuf1) == -1)
215                     fatal_error("Can't access %s\n", file1);
216             if(stat(file2, &stbuf2) == -1)
217                     fatal_error("Can't access %s\n", file2);
218             if(stbuf1.st_mtime < stbuf2.st_mtime)
219                     return(TRUE);
220             return(FALSE);
221 }
222
223 ReadCat2(fp, cat)
224 FILE    *fp;
225 int     *cat;
226 {
227             register int    i;
228             char    buf[BUFSIZ];
229
230             i = 0;
231             if(PeekChar(fp) != LBRA)
232                     return(0);
233             SkipTo(fp, LBRA);
234             while(1)
235             {
236                     GetToken(fp, buf);
237                     cat[i++] = HashSymbol(buf);
238                     if(PeekChar(fp) == RBRA)
239                             break;
240             }
```

LINE # | SOURCE TEXT

```
241         SkipTo(fp, RBRA);
242         cat[i] = -1;
243         return(1);
244 }
245
246 InstallAction2(ReducePacking, a_table, state, input, action, cat)
247 int     ReducePacking;
248 Action  **a_table;
249 int     state;
250 char    *input;
251 int     action;
252 int     *cat;
253 {
254         char    *malloc();
255         Action  *p, *q;
256
257         if(state > MAXSTATES - 1)
258                 fatal_error("LR table too large. Modify MAXSTATES.\n");
259
260         if(IsREDUCE(action) && ReducePacking)
261         {
262                 for(p = a_table[state]; p != NULL; p = p->next)
263                 {
264                         if(p->action == action)
265                         {
266                                 if(streq(input, ENDSYMBOL))
267                                         p->input = HashSymbol(ENDSYMBOL);
268                                 return;
269                         }
270                 }
271                 if(!streq(input, ENDSYMBOL))
272                         input = "*";
273         }
274
275         if((p = (Action *)malloc(sizeof(Action))) == NULL)
276                 fatal_error("No memory for action.\n");
277         p->input = InstallSymbol(input);
278         p->action = action;
279         p->next = NULL;
280 #if STOCHASTIC_LR
281         p->prob = prob;
282 #endif
283 #if TWO_LEVEL
284         if(IsSHIFT(action))
285         {
286                 int     *int_copy();
287
288                 p->cat = int_copy(cat);
289         }
290         else
291                 p->cat = NULL;
292 #endif
293         if(a_table[state] == NULL)
294                 a_table[state] = p;
295         else
296         {
297                 for(q = a_table[state]; q->next != NULL; q = q->next)
298                         ;
299                 q->next = p;
300         }
301 }
302
303 #else
304
305 ReadLR2_dummy(){}
306
307 #endif
```

LINE # SOURCE TEXT

```
/*-----------------------------------------------------------------------
 *
 *      ファイルからの読み込み
 *
 *      文法規則、LR解析表を読み込む際に用いている。
 *
 *      ATR Interpreting Telephony Research Laboratories.
 *
 *      History
 *
 *        23-Jun-1990. Created by Kenji Kita.
 *
 *-----------------------------------------------------------------------*/

#include <stdio.h>
#include <ctype.h>

#define FALSE           0
#define TRUE            1
#define LPAREN          '('
#define RPAREN          ')'

/*
 * コメント記号
 * 文法、LR解析表の各ファイルで、セミコロン(;)以下、行の終りまでは
 * コメントとみなされる。
 */
#define COMMENTCHAR     ';'

/*
 * 現在、読み込んでいるファイル中の行番号。
 */
int     lineno = 1;

/*
 * ファイル・ポインタ fp から、シンボルを1つ読み込む。
 */
GetToken(fp, s)
FILE    *fp;
char    *s;
{
        int     c;
        char    *p;

        SkipSpace(fp);
        p = s;
        while(!isspace(c = inchar(fp)) && c != LPAREN && c != RPAREN)
        {
                if(c == EOF)
                        lex_error("Unexpected EOF.\n");
                if(isuppor(c))
                        *p++ = tolower(c);
                else
                        *p++ = c;
        }
        *p = NULL;
        ungetc(c, fp);
}

/*
 * 文字 ch が見つかるまで、ファイルを読み進む。
 */
SkipTo(fp, ch)
FILE    *fp;
int     ch;
{
        register int    c;

        while((c = inchar(fp)) != ch)
        {
                if(c == EOF)
                        return(EOF);
        }
        return(ch);
}

/*
 * 空白記号（スペース、タブ等）を読みとばす。
 */
SkipSpace(fp)
FILE    *fp;
{
        int     c;

        while(isspace(c = inchar(fp)))
                ;
        if(c == EOF)
                return(EOF);
        ungetc(c, fp);
        return(c);
}

/*
 * ファイル・ポインタ fp から次に読み込まれる文字を先読みする。
 */
PeekChar(fp)
FILE    *fp;
{
        int     c;

        if(SkipSpace(fp) == EOF)
                return(EOF);
        ungetc((c = inchar(fp)), fp);
        return(c);
}

/*
 * 1文字読み込む。コメント部分は読みとばす。
 */
inchar(fp)
FILE    *fp;
{
        int     c;

        if((c = getc(fp)) == COMMENTCHAR)
        {
                while((c = getc(fp)) != '\n')
                        ;
                c = '\n';
        }
```

```
121         if(c == '\n')
122                 ++lineno;
123         return(c);
124 }
125
126 /*
127  * ファイル読み込み中に、エラーが見つかったとき、エラー・メッセージを表示。
128  * 引数は printf と同様。
129  */
130 lex_error(fmt, args)
131 char    *fmt;
132 int     args;
133 {
134         fprintf(stderr, "Error at line %d: ", lineno);
135         _doprnt(fmt, &args, stderr);
136         exit(1);
137 }
138
```

LINE # | SOURCE TEXT

```
/*----------------------------------------------------------------------*
 *
 *      エラー・メッセージの表示
 *
 *          ATR Interpreting Telephony Research Laboratories.
 *
 *          Created by Kenji Kita.
 *
 *----------------------------------------------------------------------*/

#include <stdio.h>

/*
 * エラー・メッセージを表示する．
 * 引数は printf と同様．
 */
error(fmt, args)
char    *fmt;
int     args;
{
        _doprnt(fmt, &args, stderr);
}

/*
 * エラー・メッセージを表示し，プログラムの実行を終了する．
 * 引数は printf と同様．
 */
fatal_error(fmt, args)
char    *fmt;
int     args;
{
        _doprnt(fmt, &args, stderr);
        exit(1);
}
```

LINE # SOURCE TEXT

```
/*-----------------------------------------------------------------------
 *
 *      ビームサーチによる枝刈りを行なう
 *
 *          ATR Interpreting Telephony Research Laboratories.
 *
 *      History
 *          14-Feb-1990. Created by Kenji Kita.
 *
 *-----------------------------------------------------------------------*/

#include <stdio.h>
#include "config.h"
#include "hmmlr.h"

/*
 * HMM-LRにおけるビームサーチの方法
 *
 * HMM-LR では、ビームサーチの際に、尤度の高い仮説を一定個数（ビーム幅）
 * 残し、他の仮説は枝刈りするという方式を採用している。
 * 最も単純な方法は、すべての仮説を尤度順にソートして、尤度の高いものから
 * ビーム幅の個数だけとってくるというものである。
 * しかし、この方法ではビームサーチをするごとに仮説のソートが必要になる。
 * 古いバージョンの HMM-LR ではこの方法を採用していたが、バージョン5からは、
 * 本ファイルに示されているような方式に変更した。
 * 新しい方式では、ビーム幅 n に対し、仮説の中から n 番目にある尤度を
 * まず探しだし、次にこの尤度以下の仮説を枝刈りするというものである。
 * これは、選択問題（数の集合の中で、k 番目に小さい要素を求める問題）
 * そのものであり、quicksort の分割手続きを応用した、平均して線形時間で
 * 走るアルゴリズムが知られている。本ファイルの、select_thresh() 関数が
 * これを実現する関数である。
 */

/*
 * ビームサーチの際に、threshold を決めるために、関数 select_thresh()
 * が呼び出される。select_thresh() は、quicksort と似た構造を持っており、
 * 自分自身を再帰的に呼ぶが、要素の数が QTHRESH よりし小さくなったときには、
 * 再帰呼び出しを止め、shellsort により、要素をソートする。
 * なぜ、QTHRESH を９にしてあるかというと、「データ構造とアルゴリズム」
 * （Aho, Hopcroft, Ullman の本：培風館）の本の２５８ページに
 * 「Knuth[1973] は、部分配列の大きさが９以下であったら、クイックソート
 * の中からもっと簡単なソートアルゴリズムを呼ぶように提案している。」
 * と書いてあったからである。
 */
#define QTHRESH         9

/*
 * 関数 select_thresh() の中で使われる
 * ポインタ p で指される要素と、q で指される要素の交換を行なう。
 */
#define swap(p, q)      { register double t; t = *p; *p = *q; *q = t; }

/*
 * 変数 beambuf は、ビームサーチ用のバッファであり、ビームサーチの
 * 対象となる仮説の尤度を格納するためのものである。
 * beamend は、beambuf の終りを指すポインタである。
 */
double  *beambuf, *beamend, select_thresh();

/*
 * ビームサーチ用のバッファの確保。
 */
init_beam()
{
        int     beambufsize = totalcells;

        if((beambuf = (double *)calloc(beambufsize, sizeof(double))) == NULL)
                fatal_error("No memory for beam buffer.\n");
        beamend = beambuf + beambufsize;
}

/*
 * ビームサーチを行なう。
 * セルのリスト top から尤度の大きいセルを beam 個 選び出し、再び top に代入する。
 * 枝刈りされたセルは自由セル・リストに戻される。
 */
beam_search(top, beam, cc)
Cell    **top;  /* セルのリスト */
int     beam;   /* ビーム幅 */
int     cc;     /* 現在、未使用 */
{
        register int    n;
        register double *d;
        register Cell   *cp, *p, *q;
        double  thresh;
        int     debug_flag;

        if(!*top) return;

        for(n = 0, d = beambuf, cp = *top; cp != NULL; cp = cp->next, n++)
        {
                *d++ = cp->bestprob;
                if(d > beamend)
                        break;
        }
        thresh = select_thresh(beambuf, n, beam);
        for(cp = NULL, p = *top; p != NULL; )
        {
                q = p;
                p = p->next;
                if(q->bestprob > thresh)
                {
                        q->next = freecellp;
                        freecellp = q;
                }
                else
                {
                        q->next = cp;
                        cp = q;
                }
        }
        *top = cp;
}

/*
 * 配列 v[0],v[1],...,v[n] の中から k 番目に小さい値を見つける。
 * k が n よりも大きければ、最も大きい値を返す。
 * quicksort を改良した、選択問題アルゴリズムを採用。
 */
```

LINE # | SOURCE TEXT

```
121 double  select_thresh(v, n, k)
122 double  *v;
123 int     n, k;
124 {
125         register double *left, *right, *pivot;
126         int     leftsize;
127         double  maxval();
128
129         if(k > n)
130                 return(maxval(v, n));
131         if(n <= 1)
132                 return(v[0]);
133         if(n <= QTHRESH)
134         {
135                 shellsort(v, n);
136                 return(v[k-1]);
137         }
138         left = v;
139         right = v + n - 1;
140         /*
141          * pivot の選択.
142          */
143         pivot = v;
144         swap(pivot, right);
145         pivot = right;
146         /*
147          * 分割.
148          */
149         while(left < right)
150         {
151                 while(*left < *pivot && left < right)
152                         ++left;
153                 while(*right >= *pivot && left < right)
154                         --right;
155                 if(left < right)
156                 {
157                         swap(left, right);
158                         ++left;
159                 }
160         }
161         if(right == v)
162         {
163                 swap(right, pivot);
164                 ++right;
165         }
166         if((leftsize = right - v) >= k)
167                 return(select_thresh(v, leftsize, k));
168         else
169                 return(select_thresh(right, n - leftsize, k - leftsize));
170 }
171
172 /*
173  * 配列 v[0],v[1],...,v[n] の中から最も大きい値を見つける.
174  */
175 double  maxval(v, n)
176 double  *v;
177 int     n;
178 {
179         register int    i;
180         double  max = 0.0;
181
182         for(i = 0; i < n; i++)
183         {
184                 if(v[i] > max)
185                         max = v[i];
186         }
187         return(max);
188 }
189
190 /*
191  * 配列 v[0],v[1],...,v[n] をソートする.
192  */
193 shellsort(v, n)
194 double  *v;
195 int     n;
196 {
197         register int    gap, i, j;
198         double  tmp;
199
200         for(gap = n/2; gap > 0; gap /= 2)
201         {
202                 for(i = gap; i < n; i++)
203                 {
204                         for(j = i-gap; j >= 0 && v[j] > v[j+gap]; j -= gap)
205                         {
206                                 tmp = v[j];
207                                 v[j] = v[j+gap];
208                                 v[j+gap] = tmp;
209                         }
210                 }
211         }
212 }
```

LINE # SOURCE TEXT

```
/*---------------------------------------------------------------------
 *
 *      音韻照合を行なう
 *
 *      実際のＨＭＭの確率計算は、ファイル calidm.c で行なわれる．
 *      本プログラムは、ＬＲ部分（ファイル hmmlr.c）
 *      とＨＭＭ部分（ファイル calidm.c）とのインタフェースである．
 *
 *      ATR Interpreting Telephony Research Laboratories.
 *
 *      Created by Kenji Kita and Toshiyuki Hanazawa.
 *
 *---------------------------------------------------------------------*/

#include <stdio.h>
#include <math.h>
#include "config.h"
#include "hmmlr.h"

/*
 * streq(s,t) : 文字列 s と t が等しいかを判断する．
 */
#define streq(s,t)      (*(s) == *(t) && strcmp(s,t) == 0)

#define MIN(x,y)        (((x) < (y)) ? (x) : (y))
#define MAX(x,y)        (((x) < (y)) ? (y) : (x))

/*
 * 音声入力中の最初の無音(Q1)を照合．
 */
verify_start(cp, thresh)
Cell    *cp;    /* 現在の仮説セル */
int     thresh; /* threshold */
{
        verify("Q1", 0, thresh, &cp->start, &cp->end,
                &cp->bestprob, &cp->bestpoint, cp->prob);
}

/*
 * 音声入力中の最後の無音(Q2)を照合．
 */
verify_end(cp, thresh)
Cell    *cp;    /* 現在の仮説セル */
int     thresh; /* threshold */
{
        verify("Q2", 0, thresh, &cp->start, &cp->end,
                &cp->bestprob, &cp->bestpoint, cp->prob);
}

/*
 * 音韻照合を行なう．
 * この関数の中で、start, end, bestprob, bestpoint, prob は、
 * 新たな値に更新される．
 */
verify(phone, flg_vow, thresh, start, end, bestprob, bestpoint, prob)
char    *phone;         /* 照合する音韻名 */
int     flg_vow;        /* 継続時間長パラメータの指定 */
int     thresh;         /* threshold */
int     *start;         /* 照合区間の始端 */
int     *end;           /* 照合区間の終端 */
int     *bestprob;      /* 尤度配列 prob 中の、最大正規化尤度 */
int     *bestpoint;     /* 最大正規化尤度を持つフレーム番号 */
int     *prob;          /* 尤度配列 */
{
        int     start2, end2;
        int     st, dur_f;
        Entry   *entryp, *lookup();
        IDUR    *dur_p;
        int     old_bestprob = *bestprob;

        if((entryp = lookup(phone)) == NULL)
                fatal_error("Can't lookup '%s'.\n", phone);

        dur_f = (flg_vow && entryp->num_dur > 1) ? 1 : 0;
        dur_p = entryp->duration[dur_f];

        start2 = *start + dur_p->lenmin;
        end2 = MIN(N_frames, *end + dur_p->lenmax);

        if(start2 > N_frames)
                st = 0;
        else if(streq(phone, "Q2") && end2 != N_frames)
                st = 0;
        else
        {
                cal_iprob(entryp->model, dur_p, prob,
                        *start, *end, start2, end2);

                *start = start2;
                *end = end2;

                st = best_iprob(dur_p, *start, *end, prob,
                        thresh, bestprob, bestpoint);

                if(st && First)
                        cut_prob(start, end, *bestpoint, thresh);

                ++total_verify;
        }
        *bestprob *= -1;
#if 0
        printf("verify(%s, %d, %d, %f->%f)\n", phone, *start, *end,
          (double)old_bestprob*log_henkan, (double)(*bestprob)*log_henkan);
#endif
        return(st);
}

/*
 * Beamの音韻照合区間は、連結する音韻数が増えるに従い、大きくなる．
 * cut_prob() 関数は、各フレームの正規化確率と threshold を用いて、
 * 音韻照合区間を強制的に小さくする．
 * 本関数は、First オプションが指定されたときのみ呼ばれる．
 */
cut_prob(start, end, best, thresh)
int     *start, *end, best, thresh;
{
        register        i;

        for(i = *start; i < best; i++)
                if(NormProb[i] >= thresh)
```

LINE # | SOURCE TEXT

```
121                     break;
122             *start = i;
123             for(i = *end; i > best; i--)
124                     if(NormProb[i] >= thresh)
125                     break;
126             *end = i;
127     }
```

LINE # SOURCE TEXT

```
/*-------------------------------------------------------------------
 *
 *      CPU 時間、経過時間 (elapsed time) の測定
 *
 *        ATR Interpreting Telephony Research Laboratories.
 *
 *      History
 *
 *        19-May-1988. Created by Kenji Kita.
 *
 *-------------------------------------------------------------------*/

#include <stdio.h>
#include <sys/time.h>
#include <sys/resource.h>

/*
 * CPU 時間の計測．
 * 最後にこの関数を呼んだときから使用された CPU 時間の長さ
 * (マイクロ秒単位) を返す．
 */
long    cputime()
{
        static long     pmicrosec = 0;
        long    microsec, clock();

        microsec = pmicrosec;
        pmicrosec = clock();
        return((pmicrosec - microsec)/1000);

        /*
        現在，この部分はコメントにしてあるが
        マシンによっては，以下の部分を使用すること．

        static long     psec = 0;
        long    sec;
        struct rusage   rbuf;

        getrusage(0, &rbuf);
        sec = rbuf.ru_utime.tv_sec - psec;
        psec = rbuf.ru_utime.tv_sec;
        return(sec);
        */
}

/*
 * 実行時間 (elapsed time) の計測．
 * 最後にこの関数を呼んだときからの経過時間の長さ (秒単位) を返す．
 */
long    realtime()
{
        static long     psec = 0;
        long    sec, tsec;
        long    time();

        tsec = time((long *)0);
        sec = tsec - psec;
        psec = tsec;
        return(sec);
}
```

LINE # SOURCE TEXT

```
/*---------------------------------------------------------------*
 *
 *      確率文法の各規則の適用確率のロード
 *
 *              ATR Interpreting Telephony Research Laboratories.
 *
 *      History
 *
 *              25-Aug-1988: Created by Kenji Kita.
 *
 *---------------------------------------------------------------*/

#include <stdio.h>
#include <math.h>
#include "config.h"
#include "lumnlr.h"

#if STOCHASTIC_GRAMMAR

/*
 * 非終端記号の数の上限．この値は，素数であること．
 */
#define MAXVN           1999

/*
 * 下で定義される構造体 VN の要素 rules の終りを示す．
 */
#define EOR             (-1)

/*
 * i 番目の文法規則の適用確率．
 */
#define Gprob(i)        (GrammarTable[i]->gprob)

/*
 * 文法規則の適用確率が 0 だったときの flooring の値．
 * いろいろ試してみたけど，下の値が一番よいような気がする．
 */
#define GMINPROB        1.e-4

/*
 * プロトタイプ・バージョン（単一コードブックを用いるもの）で
 * GMINPROB の値をいろいろ変えて試してみたときの文節認識率．
 */
/*#define GMINPROB      1.e-8*/         /* 75.99% */
/*#define GMINPROB      1.e-6*/         /* 76.34% */
/*#define GMINPROB      1.e-5*/         /* 76.70% */
/*#define GMINPROB      1.e-4*/         /* 76.70% */
/*#define GMINPROB      1.e-3*/         /* 77.06% */
/*#define GMINPROB      1.e-2*/         /* 75.27% */

/*
 * 各非終端記号とその非終端記号を左辺に持つ規則の
 * 対応関係を示したテーブル．
 */
typedef struct VN
{
        char    *name;  /* 非終端記号の名前 */
        int     *rules; /* name を左辺に持つ規則の番号 */
}       VN;

VN      *VN_Table[MAXVN];

/*
 * 文法の適用確率の初期化．
 */
init_prob()
{
        register int    i, j;
        int     totalrules;
        double  totalprob;
        VN      *vp;

        for(i = 0; i < MAXVN; i++)
        {
                if((vp = VN_Table[i]) == NULL)
                        continue;
                for(totalrules = 0, totalprob = 0.0, j = 0; vp->rules[j] != EOR; j++)
                {
                        totalprob += Gprob(vp->rules[j]);
                        ++totalrules;
                }
                for(j = 0; vp->rules[j] != EOR; j++)
                {
                        if(totalprob == 0)
                                Gprob(vp->rules[j]) = 0.0;
                        else
                                Gprob(vp->rules[j]) = Gprob(vp->rules[j])/totalprob;
                        /*
                         * log.
                         */
                        if(Gprob(vp->rules[j]) < GMINPROB)
                                Gprob(vp->rules[j]) = GMINPROB;
                        Gprob(vp->rules[j]) = -log10(Gprob(vp->rules[j]));
                }
        }
}

/*
 * 非終端記号 name を左辺に持つ規則の番号 ruleno を
 * VN_Table に登録する．
 */
InstallVN(name, ruleno)
char    *name;
int     ruleno;
{
        register int    i;
        int     index, size, *newrules;
        char    *p, *malloc();
        VN      *vp;

        if((vp = VN_Table[index = VNhash(name)]) == NULL)
        {
                if((vp = (VN *)malloc(sizeof(VN))) == NULL)
                        goto NO_MEMORY;
                if((p = (char *)malloc(strlen(name)+1)) == NULL)
                        goto NO_MEMORY;
                strcpy(p, name);
                if((newrules = (int *)malloc(2*sizeof(int))) == NULL)
                        goto NO_MEMORY;
```

LINE # SOURCE TEXT

```
121                 newrules[0] = ruleno;
122                 newrules[1] = EOR;
123                 vp->name = p;
124                 vp->rules = newrules;
125                 VN_Table[index] = vp;
126         }
127         else
128         {
129                 for(size = 0; vp->rules[size] != EOR; size++)
130                         ;
131                 if((newrules = (int *)malloc((size+2)*sizeof(int))) == NULL)
132                         goto NO_MEMORY;
133                 for(i = 0; vp->rules[i] != EOR; i++)
134                         newrules[i] = vp->rules[i];
135                 newrules[i++] = ruleno;
136                 newrules[i] = EOR;
137                 free((char *)vp->rules);
138                 vp->rules = newrules;
139         }
140         return(index);
141   NO_MEMORY:
142         fatal_error("No memory for non-terminal '%s'.\n", name);
143 }
144
145 /*
146  * 非終端記号 s を登録する VN_Table 上のインデックスを返す.
147  */
148 VNhash(s)
149 char    *s;
150 {
151         char    *p;
152         int     old_index, index, n;
153
154         for(index = 0, p = s; *p != NULL; )
155                 index += *p++;
156         old_index = index = index % MAXVN;
157         for(n = 0; VN_Table[index] != NULL; ++n, index = (old_index+n*n) % MAXVN)
158         {
159                 if(n != 0 && index == old_index)
160                         fatal_error("VN_Table full.\n");
161                 if(strcmp(VN_Table[index]->name, s) == 0)
162                         break;
163         }
164         return(index);
165 }
166
167 #else
168
169 StochasticGrammarDummy() {}
170
171 #endif
```

LINE # SOURCE TEXT

```
/*-----------------------------------------------------------------------
 *
 *      統計的言語モデルによる仮説の尤度計算
 *
 *          ATR Interpreting Telephony Research Laboratories.
 *
 *      History
 *
 *          15-Nov-1989, Created by Kenji Kita.
 *           3-Jul-1990, Modified.
 *
 *-----------------------------------------------------------------------*/

#include <stdio.h>
#include <math.h>
#include "config.h"
#include "hmmlr.h"
#include "ngram.h"
#include "hash.h"

/*
 * streq(s,t) : 文字列 s と t が等しいかを判断する．
 */
#define streq(s,t)      (*(s)==*(t) && strcmp(s,t)==0)

#if STOCHASTIC_GRAMMAR
/*
 * i 番目の文法規則の適用確率．
 */
#  define Gprob(i)      (GrammarTable[i]->gprob)
#endif

#if SYLLABLE_NGRAM
/*
 * N-gram モデルの値 N の定義．
 */
#  define MAXGRAM               3

/*
 * 子音の個数の上限．
 */
#  define MAXCONS               32

/*
 * 子音に後続する母音数の上限．
 */
#  define MAXMORAIDX            16

/*
 * 音節 trigram データ・ファイルの定義．
 * MORA_NGRAM の後に，以下の拡張子の付いたファイルを用いる．
 *      .1 ..... unigram count file.
 *      .2 ..... bigram count file.
 *      .3 ..... trigram count file.
 *      .sym .... symbol file.
 *      .lambda . interpolation weights file.
 *      .idx .... cons. index file.
 */
#  define MORA_NGRAM            "/ifg1/LR/Ngram/DataLvow/mora"
#endif

double  W1 = 0.0,       /* 確率文法に対する重み */
        W2 = 0.0;       /* 音節 trigram に対する重み */

/*
 * 統計的言語モデルの scaling parameter 設定．
 */
init_scaling()
{
        if(STOCHASTIC_GRAMMAR && STOCHASTIC_LR)
                fatal_error("Bad configuration.\n");
        if(STOCHASTIC_GRAMMAR && W1 > 0.0)
                printf("### USING STOCHASTIC GRAMMAR.\n");
        if(STOCHASTIC_LR && W1 > 0.0)
                printf("### USING STOCHASTIC LR PARSING.\n");
        if(SYLLABLE_NGRAM && W2 > 0.0)
                printf("### USING SYLLABLE NGRAM.\n");

        if(W1 == 0.0 && W2 == 0.0)
        {
                if(STOCHASTIC_GRAMMAR || STOCHASTIC_LR)
                        W1 = 0.04, W2 = 0.0;
                else if(SYLLABLE_NGRAM)
                        W1 = 0.0, W2 = 0.25;
        }
        printf("### W1(GrammarWeight) = %lf, W2(SyllableWeight) = %lf.\n",
                W1, W2);
        fflush(stdout);
}

/*
 * セル cp で表される認識仮説の尤度を，統計的言語モデルを用いて
 * 再計算し直す．
 * 変数 end がセットされていると，セル cp は部分的な仮説ではなく，
 * 入力音声に対するすべての音韻照合が終了していることを示す．
 */
newprob(cp, end)
Cell    *cp;
int     end;
{
        register int    i;
        double  gprob, sprob, syllable_prob();

        gprob = sprob = 0.0;

#if STOCHASTIC_GRAMMAR
        for(i = 0; i < cp->gstackptr; i++)
                gprob += Gprob(cp->gstack[i]);
#endif

#if STOCHASTIC_LR
        for(i = 0; i < cp->lstackptr; i++)
                gprob += cp->lstack[i]->prob;
printf("gprob = %lf\n", gprob);
#endif

#if SYLLABLE_NGRAM
        if(W2 > 0.0)
                sprob = syllable_prob(cp, end);
#endif
```

LINE # | SOURCE TEXT

```
121
122         cp->bestprob = (1.0 - W1 - W2)*cp->bestprob
123                         + (W1*gprob + W2*sprob)/log_henkan10;
124 }
125
126 #if SYLLABLE_NGRAM
127
128 /*
129  * 音節 trigram のデータ．
130  * Sngram[i] は (i+1)-gram のデータを格納する．
131  */
132 gtable  *Sngram[MAXGRAM];
133
134 /*
135  * 各 i-gram に対する補間の重み．
136  * Lambda[i] は i-gram に対する重みを格納する．
137  */
138 double  Lambda[MAXGRAM+1];
139
140 /*
141  * 音節の一覧表．
142  */
143 hashtable       *moratbl;
144
145 /*
146  * ConsTable[] は，子音とその子音に後続する母音を表現するのに用いる．
147  * 例えば，kya, kyu, kyo という音節があるとすると，
148  * 変数 cons に ky が，また num_mora には 3 が（ky で始まる音節数）．
149  * mora_idx[] には a, u, o に対するインデックスが入る．
150  */
151 struct
152 {
153         char    *cons;
154         int     num_mora;
155         int     mora_idx[MAXMORAIDX];
156 }       ConsTable[MAXCONS];
157
158 int     Ngram, Nsymbols, Nsamples, MaxCons = 0;
159
160 /*
161  * セル cp で表される認識仮説の尤度を，音節 trigram を用いて計算する．
162  * 変数 end がセットされていると，セル cp は部分的な仮説ではなく，
163  * 入力音声に対するすべての音韻照合が終了していることを示す．
164  */
165 double  syllable_prob(cp, end)
166 Cell    *cp;
167 int     end;
168 {
169         register int    i, j;
170         int     index, ptr, pos, len;
171         char    *lastsym, lastchar, buf[BUFSIZ];
172         double  prob, consprob, errprob = 1.e-5, delint_prob();
173
174         consprob = 0.0;
175         lastsym = SymbolTable[instacktop(cp)];
176
177         if(streq(lastsym, "pau"))
178         {
179                 len = cp->mstackptr;
180                 goto END;
181         }
182
183         if(end)                         strcat(cp->symbol, "$");
184         else if(streq(lastsym, "sy"))   strcat(cp->symbol, "sh");
185         else if(streq(lastsym, "cy"))   strcat(cp->symbol, "ch");
186         else if(streq(lastsym, "Ui"))   strcat(cp->symbol, "i");
187         else if(streq(lastsym, "Uu"))   strcat(cp->symbol, "u");
188         else                            strcat(cp->symbol, lastsym);
189
190         lastchar = cp->symbol[strlen(cp->symbol) - 1];
191
192         if(lastchar == 'a' || lastchar == 'i' || lastchar == 'u'
193         || lastchar == 'e' || lastchar == 'o' || lastchar == 'N'
194         || lastchar == 'Q' || lastchar == '$')
195         {
196                 for(pos = 0; pos < strlen(cp->symbol); )
197                 {
198                         /*
199                          * Calculate mora index.
200                          */
201                         i = search_mora(moratbl, &cp->symbol[pos], &len);
202                         if(i == -1)
203                                 return(-log10(errprob));
204                                 /*fatal_error("mora error(%s)\n", cp->symbol);*/
205
206                         mpush(i, cp);
207                         pos += len;
208
209                         /*
210                          * Calculate mora prob.
211                          */
212                         for(index = 0, i = 0; i < Ngram; i++)
213                         {
214                                 if((ptr = cp->mstackptr + i - Ngram) >= 0)
215                                         index = index*Nsymbols + cp->mstack[ptr];
216                         }
217                         prob = delint_prob(Ngram, Sngram, Lambda, Nsamples, index);
218                         cp->moraprob += -log10(prob);
219                 }
220                 cp->symbol[0] = 0x00;
221                 len = cp->mstackptr;
222         }
223         else
224         {
225                 for(index = 0, i = 0; i < Ngram - 1; i++)
226                 {
227                         if((ptr = cp->mstackptr + i - Ngram + 1) >= 0)
228                                 index = index*Nsymbols + cp->mstack[ptr];
229                 }
230                 for(i = 0; i < MaxCons; i++)
231                 {
232                         if(streq(cp->symbol, ConsTable[i].cons))
233                                 break;
234                 }
235                 if(i != MaxCons)
236                 {
237                         return(-log10(errprob));
238                         /* cellstring(1, 0, cp, buf);
239                            fatal_error("cons error(%s,%s)\n", buf, cp->symbol); */
240                 }
```

LINE # SOURCE TEXT

```
241                 for(j = 0; j < ConsTable[i].num_mora; j++)
242                 {
243                         consprob +=
244                           delint_prob(Ngram, Sngram, Lambda, Nsamples,
245                                       index + ConsTable[i].mora_idx[j]);
246                 }
247                 consprob = -log10(consprob);
248                 len = cp->mstackptr + 1;
249         }
250
251   END:
252
253         prob = (double)(cp->moraprob + consprob)/(double)len;
254
255         return(prob);
256 }
257
258 /*
259  * 音節 trigram のデータを読み込む。
260  */
261 init_sngram()
262 {
263         register int    i;
264         char    fname[80], **symtbl, *malloc();
265         int     ngram;
266         FILE    *fp;
267
268         if(W2 == 0.0)
269                 return;
270
271         /*
272          * 各 i に対し、i-gram のデータを読み込む。
273          */
274         for(i = 0; i < MAXGRAM; i++)
275         {
276                 if((Sngram[i] = (gtable *)malloc(sizeof(gtable))) == NULL)
277                         fatal_error("Can't alloc mora ngram table.\n");
278                 sprintf(fname, "%s.%d", MORA_NGRAM, i + 1);
279                 fprintf(stderr, "-> Loading %s", fname); fflush(stderr);
280                 readngram(fname, Sngram[i]);
281         }
282
283         /*
284          * 音節の一覧表を読み込む。
285          */
286         sprintf(fname, "%s.sym", MORA_NGRAM);
287         fprintf(stderr, "-> Loading %s\n", fname);
288         readsymbol(fname, &symtbl, Sngram[0]->nsymbols);
289
290         /*
291          * 読み込んだ音節の一覧表を、N-gram のデータを表す構造体の
292          * シンボル・テーブルにセット。
293          */
294         for(i = 0; i < MAXGRAM; i++)
295                 Sngram[i]->symbol = symtbl;
296
297         /*
298          * 補間の重みを読み込む。
299          */
300         sprintf(fname, "%s.lambda", MORA_NGRAM);
301         if((fp = fopen(fname, "r")) == NULL)
302                 fatal_error("Can't open %s\n", fname);
303         fscanf(fp, "%d", &ngram);
304         for(i = 0; i < ngram + 1; i++)
305         {
306                 fscanf(fp, "%lf", &Lambda[i]);
307                 printf("### Lambda[%d] = %lf\n", i, Lambda[i]);
308         }
309         fclose(fp);
310
311         /*
312          * 子音と後続母音の一覧を読み込む。
313          */
314         sprintf(fname, "%s.idx", MORA_NGRAM);
315         init_cons(fname);
316
317         /*
318          * いくつかの変数の設定。
319          */
320         Ngram = Sngram[MAXGRAM-1]->ngram;
321         Nsymbols = Sngram[MAXGRAM-1]->nsymbols;
322         Nsamples = count_samples(Sngram[0]);
323
324         /*
325          * 音節格納用のハッシュ・テーブルを作成。
326          */
327         moratbl = init_hash(Sngram[0]->nsymbols);
328         for(i = 0; i < Sngram[0]->nsymbols; i++)
329                 install_mora(moratbl, Sngram[0]->symbol[i], i);
330 }
331
332 /*
333  * ファイル fname から、子音と後続母音の一覧を読み込む。
334  */
335 init_cons(fname)
336 char    *fname;
337 {
338         register int    i, j;
339         char    cons[32], mora[32], *salloc();
340         FILE    *fp;
341
342         if((fp = fopen(fname, "r")) == NULL)
343                 fatal_error("Can't open %s\n", fname);
344         while(1)
345         {
346                 if(fscanf(fp, "%s%s", cons, mora) == EOF)
347                         break;
348                 for(i = 0; i < MaxCons; i++)
349                 {
350                         if(streq(cons, ConsTable[i].cons))
351                                 break;
352                 }
353                 if(i == MaxCons)
354                 {
355                         ConsTable[i].cons = salloc(cons);
356                         ConsTable[i].num_mora = 0;
357                 }
358                 for(j = 0; j < Sngram[0]->nsymbols; j++)
359                 {
360                         if(streq(mora, Sngram[0]->symbol[j]))
```

LINE # SOURCE TEXT

```
241                     for(j = 0; j < ConsTable[i].num_mora; j++)
242                     {
243                             consprob +=
244                               dolint_prob(Ngram, Sngram, Lambda, Nsamples,
245                                           index + ConsTable[i].mora_idx[j]);
246                     }
247                     consprob = -log10(consprob);
248                     len = cp->mstackptr + 1;
249             }
250
251     END:
252
253             prob = (double)(cp->moraprob + consprob)/(double)len;
254
255             return(prob);
256     }
257
258     /*
259      * 音節 trigram のデータを読み込む。
260      */
261     init_sngram()
262     {
263             register int    i;
264             char    fname[80], **symtbl, *malloc();
265             int     ngram;
266             FILE    *fp;
267
268             if(W2 == 0.0)
269                     return;
270
271             /*
272              * 各 i に対し、i-gram のデータを読み込む。
273              */
274             for(i = 0; i < MAXGRAM; i++)
275             {
276                     if((Sngram[i] = (gtable *)malloc(sizeof(gtable))) == NULL)
277                             fatal_error("Can't alloc mora ngram table.\n");
278                     sprintf(fname, "%s.%d", MORA_NGRAM, i + 1);
279                     fprintf(stderr, "-> Loading %s", fname); fflush(stderr);
280                     readngram(fname, Sngram[i]);
281             }
282
283             /*
284              * 音節の一覧表を読み込む。
285              */
286             sprintf(fname, "%s.sym", MORA_NGRAM);
287             fprintf(stderr, "-> Loading %s\n", fname);
288             readsymbol(fname, &symtbl, Sngram[0]->nsymbols);
289
290             /*
291              * 読み込んだ音節の一覧表を、N-gram のデータを表す構造体の
292              * シンボル・テーブルにセット。
293              */
294             for(i = 0; i < MAXGRAM; i++)
295                     Sngram[i]->symbol = symtbl;
296
297             /*
298              * 補間の重みを読み込む。
299              */
300             sprintf(fname, "%s.lambda", MORA_NGRAM);
301             if((fp = fopen(fname, "r")) == NULL)
302                     fatal_error("Can't open %s\n", fname);
303             fscanf(fp, "%d", &ngram);
304             for(i = 0; i < ngram + 1; i++)
305             {
306                     fscanf(fp, "%lf", &Lambda[i]);
307                     printf("### Lambda[%d] = %lf\n", i, Lambda[i]);
308             }
309             fclose(fp);
310
311             /*
312              * 子音と後続母音の一覧を読み込む。
313              */
314             sprintf(fname, "%s.idx", MORA_NGRAM);
315             init_cons(fname);
316
317             /*
318              * いくつかの変数の設定。
319              */
320             Ngram = Sngram[MAXGRAM-1]->ngram;
321             Nsymbols = Sngram[MAXGRAM-1]->nsymbols;
322             Nsamples = count_samples(Sngram[0]);
323
324             /*
325              * 音節格納用のハッシュ・テーブルを作成。
326              */
327             moratbl = init_hash(Sngram[0]->nsymbols);
328             for(i = 0; i < Sngram[0]->nsymbols; i++)
329                     install_mora(moratbl, Sngram[0]->symbol[i], i);
330     }
331
332     /*
333      * ファイル fname から、子音と後続母音の一覧を読み込む。
334      */
335     init_cons(fname)
336     char    *fname;
337     {
338             register int    i, j;
339             char    cons[32], mora[32], *salloc();
340             FILE    *fp;
341
342             if((fp = fopen(fname, "r")) == NULL)
343                     fatal_error("Can't open %s\n", fname);
344             while(1)
345             {
346                     if(fscanf(fp, "%s%s", cons, mora) == EOF)
347                             break;
348                     for(i = 0; i < MaxCons; i++)
349                     {
350                             if(streq(cons, ConsTable[i].cons))
351                                     break;
352                     }
353                     if(i == MaxCons)
354                     {
355                             ConsTable[i].cons = salloc(cons);
356                             ConsTable[i].num_mora = 0;
357                     }
358                     for(j = 0; j < Sngram[0]->nsymbols; j++)
359                     {
360                             if(streq(mora, Sngram[0]->symbol[j]))
```

LINE # SOURCE TEXT

```
361                             break;
362                     }
363                     if(j >= Sngram[0]->nsymbols)
364                             fatal_error("mora error(%s)\n", mora);
365                     ConsTable[i].mora_idx[ConsTable[i].num_mora++] = j;
366                     if(i == MaxCons)
367                             ++MaxCons;
368             }
369             fclose(fp);
370     #if 0
371             /*
372              * この部分、デバッグ用コード。
373              */
374             for(i = 0; i < MaxCons; i++)
375             {
376                     printf("%s: ", ConsTable[i].cons);
377                     for(j = 0; j < ConsTable[i].num_mora; j++)
378                             printf("%s ", Sngram[0]->symbol[ConsTable[i].mora_idx[j]]);
379                     printf("\n");
380             }
381             fflush(stdout);
382     #endif
383     }
384
385     /*
386      * 文字列 s を格納する領域を確保する。
387      */
388     char    *salloc(s)
389     char    *s;
390     {
391             char    *p, *malloc();
392
393             if((p = malloc(strlen(s) + 1)) == NULL)
394                     fatal_error("Can't salloc %s.\n", s);
395             strcpy(p, s);
396             return(p);
397     }
398
399     #endif
```

LINE # SOURCE TEXT

```
  1 /*---------------------------------------------------------------------
  2  *
  3  *      N-gram データの読み込み、頻度計算
  4  *
  5  *          ATR Interpreting Telephony Research Laboratories.
  6  *
  7  *      History
  8  *
  9  *        27-Oct-1989, Created by Kenji Kita.
 10  *         9-Jul-1990, Add function 'delinit_prob'.
 11  *        10-Jul-1990, Modified.
 12  *
 13  ---------------------------------------------------------------------*/
 14
 15 #include <stdio.h>
 16 #include <sys/file.h>
 17 #include <math.h>
 18 #include "config.h"
 19 #include "ngram.h"
 20
 21 #define MAXGRAM         8
 22
 23 /*
 24  * シンボルの最大長.
 25  */
 26 #define SSIZE           20
 27
 28 /*
 29  * 区切り記号.
 30  */
 31 #define SILENCE         "#"
 32
 33 /*
 34  * streq(s,t)   文字列 s と t が等しいかを判断する.
 35  */
 36 #define streq(s,t)      (*(s)==*(t) && strcmp(s,t)==0)
 37
 38 #if READ_NGRAM_DEBUG
 39 /*
 40  * デバッグ用コード.
 41  */
 42 main(argc, argv)
 43 int     argc;
 44 char    *argv[];
 45 {
 46         register int    i, j;
 47         register glist  *gp;
 48         unsigned int    n, keta[8];
 49         char    **symtbl, tmp[80], *data[80];
 50         gtable  gramtbl;
 51
 52         if(argc < 3)
 53                 fatal_error("Usage: %s Symbol-file Ngram-file\n", argv[0]);
 54         readngram(argv[2], &gramtbl);
 55         readsymbol(argv[1], &symtbl, gramtbl.nsymbols);
 56         gramtbl.symbol = symtbl;
 57         sort_symbols(&(gramtbl.symbol)[1], gramtbl.nsymbols-1);
 58         printngram(&gramtbl);
 59 }
 60
 61 #endif
 62
 63 /*
 64  * 各 i-gram に対する頻度を重み付きで足し合わせる.
 65  *
 66  *          ngram
 67  * prob = Σ    [n-gram probability] X lambda[n] の計算を行なう.
 68  *         n = 0
 69  */
 70 double  delint_prob(ngram, gramtbl, lambda, total, index)
 71 int     ngram;
 72 gtable  **gramtbl;
 73 double  *lambda;
 74 int     total;
 75 unsigned int    index;
 76 {
 77         register int    i;
 78         register glist  *gp;
 79         int     nsymbols, n, n0, index0, nc, nc0, offset;
 80         double  prob, ngramprob;
 81
 82         nsymbols = gramtbl[0]->nsymbols;
 83         prob = lambda[0]/nsymbols;
 84
 85         for(n = ngram; n > 0; n--)
 86         {
 87                 /*
 88                  * n-gram データの数をカウント.
 89                  */
 90                 for(gp = gramtbl[n-1]->table[HASH(index, gramtbl[n-1]->tablesize)];
 91                   gp != NULL; gp = gp->next)
 92                 {
 93                         if(gp->key == index)
 94                                 break;
 95                 }
 96                 nc = (gp == NULL ? 0 : (int)gp->prob);
 97
 98                 /*
 99                  * (n-1)-gram データの数をカウント.
100                  */
101                 if(n == 1)
102                         nc0 = total;
103                 else
104                 {
105                         index0 = index/nsymbols;
106                         n0 = (index0 == 0 ? 0 : n-2);
107                         for(gp = gramtbl[n0]->table[HASH(index0, gramtbl[n0]->tablesize)];
108                           gp != NULL; gp = gp->next)
109                         {
110                                 if(gp->key == index0)
111                                         break;
112                         }
113                         nc0 = (gp == NULL ? 0 : (int)gp->prob);
114                 }
115
116                 /*
117                  * n-gram の頻度を重み付きで足し合わせる.
118                  */
119                 ngramprob = (nc0 == 0 ? 0.0 : (double)nc/nc0);
120                 prob += lambda[n]*ngramprob;
```

LINE # | SOURCE TEXT

```
121
122                     /*
123                      * (n-1)-gram データの gramtbl でのインデックスの計算.
124                      */
125                     for(offset = nsymbols, i = 1; i < n - 1; i++)
126                             offset *= nsymbols;
127                     index = index % offset;
128             }
129             return(prob);
130     }
131
132     /*
133      * 学習データ中の総音節数のカウント.
134      */
135     count_samples(gramtbl)
136     gtable  *gramtbl;
137     {
138             register int    i;
139             register glist  *gp;
140             int     n;
141
142             for(n = 0, i = 0; i < gramtbl->nsymbols; i++)
143             {
144                     for(gp = gramtbl->table[HASH(i, gramtbl->tablesize)];
145                       gp != NULL; gp = gp->next)
146                     {
147                             if(gp->key == i)
148                                     n += (int)gp->prob;
149                     }
150             }
151             return(n);
152     }
153
154     /*
155      * ファイル fname から N-gram のデータを読み込み gramtbl に格納.
156      */
157     readngram(fname, gramtbl)
158     char    *fname;
159     gtable  *gramtbl;
160     {
161             register int    i;
162             int     fd, index, count = 0, pcount;
163             unsigned int    key;
164             float   prob;
165             glist   *gp;
166
167             if((fd = open(fname, O_RDONLY)) < 0)
168                     fatal_error("Can't open %s\n", fname);
169             read(fd, &gramtbl->ngram, sizeof(int));
170             read(fd, &gramtbl->nsymbols, sizeof(int));
171             read(fd, &gramtbl->tablesize, sizeof(int));
172     #if DEC_TO_IEEE
173             long_dec_to_ieee(&gramtbl->ngram, 1);
174             long_dec_to_ieee(&gramtbl->nsymbols, 1);
175             long_dec_to_ieee(&gramtbl->tablesize, 1);
176     #endif
177
178     #if 0
179             printf("ngram = %d, nsymbols = %d, tablesize = %d\n",
180                     gramtbl->ngram, gramtbl->nsymbols, gramtbl->tablesize);
181     #endif
182             if((gramtbl->table =
183               (glist **)malloc(gramtbl->tablesize * sizeof(glist *))) == NULL)
184                     fatal_error("Can't allocate ngram table.\n");
185             for(i = 0; i < gramtbl->tablesize; i++)
186                     gramtbl->table[i] = NULL;
187             while(1)
188             {
189                     if(read(fd, &index, sizeof(int)) == 0)
190                             break;
191     #if DEC_TO_IEEE
192                     long_dec_to_ieee(&index, 1);
193     #endif
194                     if((gp = (glist *)malloc(sizeof(glist))) == NULL)
195                             fatal_error("Can't allocate glist.\n");
196                     read(fd, &gp->key, sizeof(int));
197                     read(fd, &gp->prob, sizeof(float));
198     #if DEC_TO_IEEE
199                     long_dec_to_ieee(&gp->key, 1);
200                     float_dec_to_ieee(&gp->prob, 1);
201     #endif
202                     gp->next = gramtbl->table[index];
203                     gramtbl->table[index] = gp;
204                     ++count;
205     #if 1
206                     if(count != 0 && count % 1000 == 0)
207                     {
208                             fprintf(stderr, "."); fflush(stderr);
209                             pcount = count;
210                     }
211     #endif
212             }
213     #if 1
214             /*if(count != pcount)*/
215             {
216                     fprintf(stderr, "\n");
217             }
218     #endif
219             close(fd);
220     }
221
222     /*
223      * 音節の一覧表をファイル fname から読み込み、symtbl に格納する.
224      */
225     readsymbol(fname, symtbl, size)
226     char    *fname, ***symtbl;
227     int     size;   /* 音節の総数 */
228     {
229             int     nsymbols = 0;
230             char    buf[SSIZE], *p;
231             FILE    *fp;
232
233             if((*symtbl = (char **)malloc(size * sizeof(char *))) == NULL)
234                     fatal_error("Can't allocate symbol table. (size=%d)\n", size);
235             strcpy(buf, SILENCE);
236             if((p = (char *)malloc(strlen(buf)+1)) == NULL)
237                     fatal_error("No memory for symbol '%s'.\n", buf);
238             strcpy(p, buf);
239             (*symtbl)[nsymbols++] = p;
240             if((fp = fopen(fname, "r")) == NULL)
```

LINE # | SOURCE TEXT

```
241                 fatal_error("Can't open %s\n", fname);
242         while(1)
243         {
244                 if(fgets(buf, sizeof(buf), fp) == NULL)
245                         break;
246                 if(strlen(buf) == 1)
247                         continue;
248                 buf[strlen(buf)-1] = NULL;
249                 if(buf[0] == '@')
250                         continue;
251                 if((p = (char *)malloc(strlen(buf)+1)) == NULL)
252                         fatal_error("No memory for symbol '%s'.\n", buf);
253                 strcpy(p, buf);
254                 (*symtbl)[nsymbols++] = p;
255                 if(nsymbols > size)
256                         fatal_error("Too small symbol table size.\n");
257         }
258         fclose(fp);
259         return(nsymbols);
260 }
261
262 /*
263  * symtbl 上にある size 個の文字列をソートする．
264  */
265 sort_symbols(symtbl, size)
266 char    **symtbl;
267 int     size;
268 {
269         register int    i, j, k;
270         char    *t;
271
272         for(k = size/2; k > 0; k /= 2)
273         {
274                 for(i = k; i < size; i++)
275                 {
276                         for(j = i - k; j >= 0; j -= k)
277                         {
278                                 if(strlen(symtbl[j]) >= strlen(symtbl[j+k]))
279                                         break;
280                                 t = symtbl[j];
281                                 symtbl[j] = symtbl[j+k];
282                                 symtbl[j+k] = t;
283                         }
284                 }
285         }
286 }
287
288 /*
289  * デバッグ用コード．未使用．
290  */
291 printngram(gramtbl)
292 gtable  *gramtbl;
293 {
294         register int    i, j;
295         register glist  *gp;
296         unsigned int    n, keta[8];
297         char    tmp[80];
298
299         for(i = 0; i < gramtbl->tablesize; i++)
300         {
301                 for(gp = gramtbl->table[i]; gp != NULL; gp = gp->next)
302                 {
303                         for(n = gp->key, j = gramtbl->ngram - 1; j >= 0; j--)
304                         {
305                             keta[j] = n % gramtbl->nsymbols;
306                             n /= gramtbl->nsymbols;
307                         }
308                         for(tmp[0] = NULL, j = 0; j < gramtbl->ngram; j++)
309                         {
310                             strcat(tmp, gramtbl->symbol[keta[j]]);
311                             strcat(tmp, " ");
312                         }
313                         printf(" %-14s -> prob: %f,  key: %u\n",
314                                 tmp, gp->prob, gp->key);
315                 }
316         }
317 }
```

LINE # SOURCE TEXT

```
/*-----------------------------------------------------------------------
 *
 *      音節検索のためのハッシュ関数
 *
 *       ATR Interpreting Telephony Research Laboratories.
 *
 *      History
 *
 *        4-Jul-1990. Created by Kenji Kita.
 *
 *-----------------------------------------------------------------------*/

#include <stdio.h>
#include "hash.h"

/*
 * streq(s,t) : 文字列 s と t が等しいかを判断する．
 */
#define streq(s,t)      (*(s) == *(t) && strcmp(s,t) == 0)

/*
 * ハッシュ・テーブルの初期化．
 * 大きさが size であるハッシュ・テーブルを作り，そのテーブルへの
 * ポインタを返す．
 */
hashtable       *init_hash(size)
int     size;
{
        register int    i;
        hashtable       *tbl;
        char    *malloc();

        if((tbl = (hashtable *)malloc(sizeof(hashtable))) == NULL)
                fatal_error("Can't alloc hashtable.\n");
        if((tbl->entrytable = (entry **)malloc(sizeof(entry *)*size)) == NULL)
                fatal_error("Can't alloc entrytable.\n");
        for(i = 0; i < size; i++)
                tbl->entrytable[i] = NULL;
        tbl->size = size;
        return(tbl);
}

/*
 * ハッシュ・テーブル tbl から，文字列 key の接頭語となる音節を見つける．
 * len には，ハッシュ・テーブル中に見つかった音節の長さが入れられる．
 * また，見つかった音節の moratbl 中でのインデックスを返す．
 */
search_mora(tbl, key, len)
hashtable       *tbl;
char    *key;
int     *len;
{
        register entry  *p;

        for(p = tbl->entrytable[MHASH(key, tbl->size)]; p != NULL; p = p->next)
        {
                if(prefix(key, p->key))
                {
                        *len = strlen(p->key);
                        return(p->data);
                }
        }
        return(-1);
}

/*
 * ハッシュ・テーブル tbl に文字列 key を登録する．
 * 実際には，key は音節名であり，data は音節の moratbl 上でのインデックスである．
 */
install_mora(tbl, key, data)
hashtable       *tbl;
char    *key;
int     data;
{
        register entry  *p, *q;
        entry   *new;
        char    *malloc();
        int     n;

        for(new = tbl->entrytable[MHASH(key, tbl->size)]; new != NULL; new = new->next)
        {
                if(streq(key, new->key))
                        break;
        }
        if(new)
        {
                error("%s: already exists.\n", key);
                return;
        }
        if((new = (entry *)malloc(sizeof(entry))) == NULL)
                fatal_error("No memory. Can't install %s.\n", key);
        if((new->key = (char *)malloc(strlen(key) + 1)) == NULL)
                fatal_error("No memory. Can't install %s.\n", key);
        strcpy(new->key, key);
        new->data = data;

        for(q = NULL, p = tbl->entrytable[n = MHASH(key, tbl->size)]; p != NULL; p = p->next)
        {
                if(strlen(key) >= strlen(p->key))
                        break;
                q = p;
        }
        if(!q)
        {
                new->next = tbl->entrytable[n];
                tbl->entrytable[n] = new;
        }
        else
        {
                new->next = q->next;
                q->next = new;
        }
}

/*
 * 文字列 s2 が，文字列 s1 の接頭語であるかどうかを調べる．
 * 接頭語であれば，1 を返す．
 */
prefix(s1, s2)  /* 's2' is a prefix of 's1'. */
unsigned char   *s1, *s2;
```

```
LINE #  SOURCE TEXT
  121   {
  122           while(*s2)
  123           {
  124                   if(*s1++ != *s2++)
  125                           return(0);
  126           }
  127           return(1);
  128   }
  129
  130   /*
  131    * 未使用.
  132    */
  133   hash_string(s, n)
  134   register char   *s;
  135   int     n;
  136   {
  137           register int    v = 0;
  138
  139           while(*s)
  140                   v += *s++;
  141           return(v%n);
  142   }
```

LINE # SOURCE TEXT

```
/*=============================================================================
 *
 *      2段階ＬＲの文節同ＬＲパーザ
 *
 *          ATR Interpreting Telephony Research Laboratories.
 *
 *      History
 *
 *          23-Apr-1990, Created by Kenji Kita.
 *          12-Apr-1991, Modified.
 *
 *===========================================================================*/

#include <stdio.h>
#include "config.h"

#if TWO_LEVEL

#include "hmmlr.h"

/*
 * coll.h で定義されている値を無効にし、新たに定義し直す。
 * 以下での定義は、文節同LRパーザ用のものである。
 */
#undef TOTALCELLS
#undef LR_STACKSIZE
#undef IN_STACKSIZE
#undef G_STACKSIZE
#undef WORD_DELIMITER

#define TOTALCELLS      1024
#define LR_STACKSIZE    100
#define IN_STACKSIZE    200
#define G_STACKSIZE     100
#define WORD_DELIMITER  (-1)

#undef stacktop
#undef push
#undef npop
#undef clrstack
#undef instacktop
#undef inpush
#undef clrinstack
#undef gpush
#undef clrgstack

#define stacktop(cp)    cp->stack[cp->stackptr-1]
#define push(n, cp)     cp->stack[(cp->stackptr)++] = n
#define npop(n, cp)     cp->stack[(cp->stackptr -= n)]
#define clrstack(cp)    cp->stackptr = 0
#define instacktop(cp)  cp->instack[cp->instackptr-1]
#define inpush(n, cp)   cp->instack[(cp->instackptr)++] = n
#define clrinstack(cp)  cp->instackptr = 0
#define gpush(n, cp)    cp->gstack[(cp->gstackptr)++] = n
#define clrgstack(cp)   cp->gstackptr = 0

/*
 * streq(s,t)   文字列 s と t が等しいかを判断する。
 */
#define streq(s,t)      (*(s) == *(t) && strcmp(s,t) == 0)

extern double   *beambuf,       /* ビームサーチ用 (beam.c 参照) */
                *beamend,       /*  同    上  */
                select_thresh(),/*  同    上  */

extern double   log_henkan;     /* hmmlr.h 参照 */
extern int      Nbest;          /*  同    上  */
extern Rule     *Gtable[];      /*  同    上  */
extern Action   *Atable[];      /*  同    上  */

extern int      topcat[];       /* hmmlr.c 参照 */

int     s_globalbeam,           /* 文節同LR用のビーム幅 */
        s_localbeam;            /* 文節同LR用の局所的ビーム幅 */
int     phrase_no;              /* 何番目の文節か？ */
int     Nbest2;                 /* 文の出力候補数 */

/*
 * ２段階LRで、文の認識仮説のためのセル。
 * 名前は Scell と違うが、名変数の意味は Coll と同様。
 * ファイル coll.h 参照のこと。
 */
typedef struct Scell
{
        int     stack[LR_STACKSIZE];
        char    stackptr;
        int     instack[IN_STACKSIZE];
        unsigned char   instackptr;
        int     gstack[G_STACKSIZE];
        char    gstackptr;

        double  score;  /* 文のスコア */

        struct Scell    *next;
}       Scell;

Scell   *s_cellstart,
        *s_cellend,
        *s_celltop,
        *s_newcelltop,
        *s_freecellp,
        *s_acceptedcellp,
        *s_localcellp;

/*
 * ２段階LRのメイン・ルーチン。
 */
LRctrl_main(fp)
FILE    *fp;
{
        s_globalbeam = globalbeam;
        s_localbeam = localbeam;
        Nbest2 = Nbest * 2;

        s_init_cell();

        while(1)
        {
                if(LRctrl(fp) == EOF)
                        break;
```

LINE # SOURCE TEXT

```
121         }
122 }
123
124 /*
125  * ２段階LRにより，１文を認識する．
126  * 基本的な処理の流れは，hmmlr.cと同様であるのでいちいち説明しない．
127  * 特徴的な箇所だけ説明を付ける．
128  */
129 LRctrl(fp)
130 FILE    *fp;
131 {
132         register int    i, j;
133         register Scell  *cp, *cpp, *newcp;
134         register Action *ap;
135         int     state, newstate, action, count, offs, other_action, m, st;
136         char    buf[BUFSIZ];
137         Scell   *s_getcopy();
138
139         register Cell   *cp2;
140         register Action *ap2;
141         int     prediction;
142         Cell    *getcopyl();
143
144         phrase_no = 0;
145         st = TRUE;
146         s_reinitcell();
147         while(1)
148         {
149             /*
150              * 還元動作．
151              */
152             while(s_celltop)
153             {
154                 cp = s_celltop;
155                 s_celltop = s_celltop->next;
156                 other_action = 0;
157                 state = stacktop(cp);
158                 for(ap = Atable[state]; ap != NULL; ap = ap->next)
159                 {
160                     action = ap->action;
161                     if(IsACCEPT(action) || IsSHIFT(action))
162                         other_action++;
163                     else if(IsREDUCE(action))
164                     {
165                         newcp = s_getcopy(cp);
166                         m = Gtable[REDUCE(action)]->length;
167                         npop(m, newcp);
168                         newstate = GotoState(Atable, stacktop(newcp),
169                             Gtable[REDUCE(action)]->lhs);
170                         push(newstate, newcp);
171                         if(instacktop(newcp) != WORD_DELIMITER)
172                             inpush(WORD_DELIMITER, newcp);
173                         if(newcp->instackptr >= IN_STACKSIZE)
174                         {
175                             error("LRctrl: Input stack overflow\n");
176                             return;
177                         }
178                         gpush(REDUCE(action), newcp);
179                         newcp->next = s_celltop;
180                         s_celltop = newcp;
181                     }
182                 }
183                 if(other_action)
184                 {
185                     cp->next = s_newcelltop;
186                     s_newcelltop = cp;
187                 }
188                 else
189                 {
190                     cp->next = s_freecellp;
191                     s_freecellp = cp;
192                 }
193                 s_beam_search(&s_celltop, s_globalbeam, 'R');
194             }
195             s_celltop = s_newcelltop;
196             s_newcelltop = NULL;
197             s_beam_search(&s_celltop, s_globalbeam, 'R');
198
199             count = 0;
200
201             if(st == FALSE) fatal_error("shift error.\n");
202
203             /*
204              * １文節の音声データ入力．
205              */
206             st = Read_Fuzzy(fp);
207             if(st == EOF) goto end;
208             if(st == TRUE)
209             {
210                 ++phrase_no;
211                 cal_frame_prob();
212                 reinitcell();
213                 clear_topcat();
214             }
215
216             prediction = 0;
217             for(cp = s_celltop; cp != NULL; cp = cp->next)
218             {
219                 offs = 0;
220                 s_localcellp = NULL;
221                 state = stacktop(cp);
222                 for(ap = Atable[state]; ap != NULL; ap = ap->next)
223                 {
224                     action = ap->action;
225                     /*
226                      * 受理動作．
227                      */
228                     if(IsACCEPT(action))
229                     {
230                         if(st == FALSE)
231                         {
232                             newcp = s_getcopy(cp);
233                             newcp->next = s_acceptedcellp;
234                             s_acceptedcellp = newcp;
235                         }
236                     }
237                     /*
238                      * 移動動作．
239                      */
240                     else if(IsSHIFT(action))
```

LINE # SOURCE TEXT

```
241                     {
242                         if(st == TRUE)
243                         {
244                             sprintf(buf, "<%s>", SymbolTable[ap->input]);
245                             topcat[HashSymbol(buf)] = 1;
246                             ++prediction;
247                         }
248                     }
249                 }
250             }
251
252             if(st == TRUE && prediction)
253             {
254                 /*
255                  * HMM-LR を呼び出し，文節認識を行なう．
256                  */
257                 hmmlr();
258
259                 while(s_celltop)
260                 {
261                     cp = s_celltop;
262                     s_celltop = s_celltop->next;
263                     offs = 0;
264                     state = stacktop(cp);
265                     for(ap = Atable[state]; ap != NULL; ap = ap->next)
266                     {
267                         action = ap->action;
268                         if(IsSHIFT(action))
269                         {
270                             for(i = 0, cp2 = acceptedcellp;
271                               i < Nbest2 && cp2 != NULL; cp2 = cp2->next, i++)
272                             {
273                                 sprintf(buf, "<%s>", SymbolTable[ap->input]);
274                                 if(HashSymbol(buf) == cp2->topcat)
275                                 {
276                                     newcp = s_getcopy(cp);
277                                     for(j = 0; j < cp2->instackptr; j++)
278                                         inpush(cp2->instack[j], newcp);
279                                     newcp->score
280                                         += (double)cp2->bestprob * log_henkan;
281                                     newstate = SHIFT(action);
282                                     push(newstate, newcp);
283                                     newcp->next = s_localcellp;
284                                     s_localcellp = newcp;
285                                     ++offs;
286                                 }
287                             }
288                         }
289                     }
290                     if(offs)
291                     {
292                         if(offs > s_localbeam)
293                             s_beam_search(&s_localcellp, s_localbeam, 'L');
294
295                         for(cpp = s_localcellp; cpp->next != NULL; cpp = cpp->next)
296                             ++count;
297                         ++count;
298                         cpp->next = s_newcelltop;
299                         s_newcelltop = s_localcellp;
300                         s_localcellp = NULL;
301                     }
302                     cp->next = s_freecellp;
303                     s_freecellp = cp;
304                 }
305             }
306             s_celltop = s_newcelltop;
307             s_newcelltop = NULL;
308
309             if(count == 0)
310                 break;
311
312             if(count > s_globalbeam)
313                 s_beam_search(&s_celltop, s_globalbeam, 'G');
314         }
315     end:
316         if(st == EOF)
317             return(EOF);
318         /*
319          * 未処理の音声データあり，文の認識失敗．残念！
320          */
321         if(st == TRUE)
322         {
323             while(getc(fp) != '!')  ;
324             while(getc(fp) != '\n') ;
325         }
326
327         s_rm_dupl(&s_acceptedcellp);
328
329     printf("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n");
330     if(!s_acceptedcellp || st == TRUE) printf(" NO RECOGNITION CANDIDATES.\n");
331     show_scells(&s_acceptedcellp, Nbest2);
332     printf("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n\n");
333     fflush(stdout);
334     return(TRUE);
335 }
336
337 /*
338  * 文の認識候補を max 個 出力．
339  */
340 show_scells(top, max)
341 Scell   **top;
342 int     max;
343 {
344         register int    i, j;
345         Scell   *cp;
346
347         s_sortcell(top);
348         for(cp = *top, i = 0; cp != NULL; cp = cp->next, i++)
349         {
350                 if(i >= max)
351                         break;
352                 printf("%3d: ", i + 1);
353                 show_cell(cp);
354                 printf(" (score = %f)", cp->score);
355                 printf("\n");
356                 fflush(stdout);
357         }
358 }
359
360 /*
```

LINE # SOURCE TEXT

```
361  * 1つの文仮説を出力.
362  */
363 show_cell(cp)
364 Scell   *cp;
365 {
366         register int    i;
367         char    buf[BUFSIZ];
368
369         buf[0] = NULL;
370         if(cp->instackptr == 0)
371                 strcat(buf, "[Empty]");
372         else
373         {
374                 for(i = 0; i < cp->instackptr; i++)
375                 {
376                         if(i + 1 == cp->instackptr)
377                                 break;
378                         if(cp->instack[i] == WORD_DELIMITER)
379                                 strcat(buf, "-");
380                         else
381                         {
382                             if(streq(SymbolTable[cp->instack[i]], "Ui"))
383                                 strcat(buf, "i");
384                             else if(streq(SymbolTable[cp->instack[i]], "Uu"))
385                                 strcat(buf, "u");
386                             else
387                                 strcat(buf, SymbolTable[cp->instack[i]]);
388                         }
389                 }
390         }
391         printf("%-35s", buf);
392 }
393
394 /*
395  * 以下の関数群は、文仮説セルに対する操作を行なうものである.
396  * これらの関数は、cell.c で定義されているものとほぼ同様.
397  * ファイル cell.c 参照のこと.
398  */
399 s_init_cell()
400 {
401         char    *calloc( );
402
403         if((s_cellstart = (Scell*)calloc(totalcells, sizeof(Scell))) == NULL)
404                 fatal_error("No memory for cells.\n");
405 }
406
407 s_reinitcell()
408 {
409         register int    i;
410         register Scell  *p;
411         Scell   *s_getcell();
412
413         s_cellend = s_cellstart + totalcells;
414         for(p = s_cellstart; p < s_cellend - 1; p++)
415                 p->next = p + 1;
416         p->next = NULL;
417
418         s_freecellp = s_cellstart;
419         s_newcelltop = NULL;
420         s_acceptedcellp = NULL;
421
422         s_celltop = s_getcell();
423         s_celltop->score = 0.0;
424         s_celltop->next = NULL;
425         clrstack(s_celltop);
426         clrinstack(s_celltop);
427         clrgstack(s_celltop);
428
429         push(0, s_celltop);
430 }
431
432 Scell   *s_getcell()
433 {
434         Scell   *p;
435
436         if(s_freecellp == NULL)
437                 s_GC();
438         p = s_freecellp;
439         s_freecellp = s_freecellp->next;
440         return(p);
441 }
442
443 Scell   *s_getcopy(p)
444 Scell   *p;
445 {
446         register int    i;
447         Scell   *q;
448
449         if(s_freecellp == NULL)
450                 s_GC();
451         q = s_freecellp;
452         s_freecellp = s_freecellp->next;
453
454         bcopy((char *)p, (char *)q, sizeof(Scell));
455         return(q);
456 }
457
458 s_GC()
459 {
460         int     beam;
461
462         beam = s_globalbeam;
463  start:
464         s_beam_search(&s_newcelltop, beam, 'G');
465         if(!s_freecellp)
466         {
467                 s_beam_search(&s_acceptedcellp, Nbest2, 'G');
468                 if(s_freecellp == NULL)
469                 {
470                         --beam;
471                         if(!beam)
472                                 fatal_error("Scell exhausted.\n");
473                         goto start;
474                 }
475         }
476 }
477
478 s_rm_dupl(top)
479 Scell   **top;
480 {
```

```
LINE #  SOURCE TEXT
481         register int    i;
482         register Scell  *p, *q, *r, *cp;
483         int     ok;
484
485         for(cp = NULL, p = *top; p != NULL; )
486         {
487                 r = p;
488                 p = p->next;
489                 ok = 1;
490                 for(q = cp; q != NULL; q = q->next)
491                 {
492                         if(r->instackptr != q->instackptr)
493                                 continue;
494                         for(i = 0; i < r->instackptr; i++)
495                         {
496                                 if(r->instack[i] != q->instack[i])
497                                         break;
498                         }
499                         if(i == r->instackptr)
500                         {
501                                 ok = 0;
502                                 break;
503                         }
504                 }
505                 if(!ok)
506                 {
507                         q->score = (r->score < q->score ? r->score : q->score);
508                         r->next = s_freecellp;
509                         s_freecellp = r;
510                 }
511                 else
512                 {
513                         r->next = cp;
514                         cp = r;
515                 }
516         }
517         *top = cp;
518 }
519
520 s_sortcell(cell)
521 Scell   **cell;
522 {
523 #define insert(cp, first, last)                         \
524                         { if(!first) first = cp;        \
525                           else last->next = cp;         \
526                           last = cp; }
527
528         Scell   *cp = *cell, *lowf, *lowl, *midf, *midl, *highf, *highl;
529
530         if(cp == NULL)
531                 return;
532         lowf = midf = highf = NULL;
533         insert(cp, midf, midl);
534         for(cp = cp->next; cp != NULL; cp = cp->next)
535         {
536                 if(cp->score < midf->score)
537                         insert(cp, lowf, lowl)
538                 else if(cp->score == midf->score)
539                         insert(cp, midf, midl)
540                 else
541                         insert(cp, highf, highl)
542         }
543         if(lowf != NULL)
544         {
545                 lowl->next = NULL;
546                 s_sortcell(&lowf);
547                 for(cp = lowf; cp->next != NULL; cp = cp->next)
548                         ;
549                 cp->next = midf;
550                 cp = lowf;
551         }
552         else
553                 cp = midf;
554         if(highf != NULL)
555                 highl->next = NULL;
556         s_sortcell(&highf);
557         midl->next = highf;
558         *cell = cp;
559 }
560
561 /*
562  * 文仮説候補に対するビームサーチ．
563  * 関数 beam_search() とほぼ同様．
564  * ファイル beam.c 参照のこと．
565  */
566 s_beam_search(top, beam, cc)
567 Scell   **top;
568 int     beam;
569 int     cc;
570 {
571         register int    n;
572         register double *d;
573         register Scell  *cp, *p, *q;
574         double  thresh;
575
576         if(!*top) return;
577         for(n = 0, d = beambuf, cp = *top; cp != NULL; cp = cp->next, n++)
578         {
579                 *d++ = cp->score;
580                 if(d > beamend)
581                         break;
582         }
583         thresh = select_thresh(beambuf, n, beam);
584         for(cp = NULL, p = *top; p != NULL; )
585         {
586                 q = p;
587                 p = p->next;
588                 if(q->score > thresh)
589                 {
590                         q->next = s_freecellp;
591                         s_freecellp = q;
592                 }
593                 else
594                 {
595                         q->next = cp;
596                         cp = q;
597                 }
598         }
599         *top = cp;
600 }
```

LINE # | SOURCE TEXT

```
601
602  /*
603   * 変数 topcat[] は、文節間LRが予測した文節カテゴリを保持する．
604   * 関数 clear_topcat() は、この変数をクリアする．
605   */
606  clear_topcat()
607  {
608          register int    i;
609
610          for(i = 0; i < MAXSYMBOLS; i++)
611                  topcat[i] = 0;
612  }
613
614  /*
615   * 変数 pcat で示される文節カテゴリが、予測されているかどうか
616   * チェックする．
617   */
618  predicted(pcat)
619  int     *pcat;
620  {
621          register int    *p;
622
623          if(pcat == NULL)
624                  fatal_error("No prediction.\n");
625          for(p = pcat; *p != -1; p++)
626          {
627                  if(topcat[*p] == 1)
628                          return(1);
629          }
630          return(0);
631  }
632
633  #else
634
635  LRctrl_main(){}
636
637  #endif
```

LINE # SOURCE TEXT

```
/*---------------------------------------------------------------------*
 *
 *      整数配列（ポインタ）の操作
 *
 *        本プログラムは、２段階ＬＲでのみ用いられる
 *        文節内ＬＲ解析表に到達可能な文法カテゴリを登録するのに用いる。
 *
 *          ATR Interpreting Telephony Research Laboratories.
 *
 *      Edit History
 *
 *        29-Jun-1991, Created by Kenji Kita.
 *
 *---------------------------------------------------------------------*/

#include <stdio.h>
#include "config.h"
#include "LR.h"

#if TWO_LEVEL

#define END     (-1)

/*
 * ファイル・ポインタ fp から、文字列を読み込み、読み込んだ文字列の
 * SymbolTable でのインデックスを整数配列 ptr に追加する。
 */
ReadCat(fp, ptr)
FILE    *fp;
int     **ptr;
{
        char    buf[BUFSIZ];

        *ptr = NULL;
        while(fscanf(fp, "%s", buf) != EOF)
        {
                int_add(HashSymbol(buf), ptr);
        }
}

/*
 * デバッグ用コード、未使用。
 */
PrintCat(fp, ptr)
FILE    *fp;
int     *ptr;
{
        register int    *p;

        if(!ptr)
                return;
        fprintf(fp, " [ ");
        for(p = ptr; *p != END; p++)
        {
                fprintf(fp, "%s", SymbolTable[*p]);
                if(*(p+1) != END)
                        fprintf(fp, " ");
        }
        fprintf(fp, " ] ");
}

/*
 * 整数配列 ptr に、ptr2 の各要素を追加する。
 */
int_union(ptr, ptr2)
int     **ptr, *ptr2;
{
        register int    *p;

        if(!ptr2)
                return;
        for(p = ptr2; *p != END; p++)
                int_add(*p, ptr);
}

/*
 * 整数配列 ptr に整数 n を追加する。
 */
int_add(n, ptr)
int     n, **ptr;
{
        register int    *p, *q;
        int     *ptr2, size;

        size = 0;
        if(*ptr)
        {
                for(p = *ptr; *p != END; p++, ++size)
                {
                        if(*p == n)
                                return;
                }
        }
        if((ptr2 = (int *)malloc((size+2) * sizeof(int))) == NULL)
                return(NULL);
        q = ptr2;
        if(*ptr)
        {
                for(p = *ptr; *p != END; p++, q++)
                        *q = *p;
        }
        *q++ = n;
        *q = END;
        if(*ptr)
                free((char *)*ptr);
        *ptr = ptr2;
}

/*
 * 整数配列 ptr の中に、整数 n が入っているか調べる。
 */
int_exist(n, ptr)
int     n, *ptr;
{
        register int    *p;

        if(!ptr)
                return(0);
        for(p = ptr; *p != END; p++)
        {
```

LINE # | SOURCE TEXT

```
121                 if(*p == n)
122                         return(1);
123         }
124         return(0);
125 }
126
127 /*
128  * 整数配列 ptr をコピーして、新しい整数配列を作り、
129  * それへのポインタを返す。
130  */
131 int     *int_copy(ptr)
132 int     *ptr;
133 {
134         register int    *p, *q;
135         int     *ptr2, size;
136
137         if(!ptr)
138                 return(NULL);
139         size = 0;
140         for(p = ptr; *p != END; p++)
141                 ++size;
142         if((ptr2 = (int *)malloc((size+1) * sizeof(int))) == NULL)
143                 return(NULL);
144         for(p = ptr, q = ptr2; *p != END; *q++ = *p++)
145                 ;
146         *q = END;
147         return(ptr2);
148 }
149
150 /*
151  * デバック用コード：未使用：
152  */
153 int_print(ptr)
154 int     *ptr;
155 {
156         register int    *p;
157
158         for(p = ptr; *p != END; p++)
159                 printf("%d ", *p);
160         printf("\n");
161 }
162
163 #else
164
165 int_op_dummy(){}
166
167 #endif
```

LINE # SOURCE TEXT

```
  1 #include "config.h"
  2 #include "imhmm.h"
  3 #include "idur.h"
  4 #include "log.h"
  5
  6 #define  MIN( x, y )    ( ( (x) < (y) ) ? (x) : (y) )
  7 #define  MAX( x, y )    ( ( (x) < (y) ) ? (y) : (x) )
  8
  9 extern int      *Alpha[], NormProb[];
 10
 11
 12 cal_iprob( hmm, dur, prob, start1, end1, start2, end2 )
 13 IMHMM    *hmm;
 14 IDUR     *dur;
 15 int      *prob, start1, end1, start2, end2;
 16 {
 17     int      i, j, jj, k, tx, final;
 18     int      ax, rarc, index, roop_prob;
 19
 20 #if 0
 21   printf( "start1= %d end1= %d  start2= %d end2= %d\n",
 22                                     start1, end1, start2, end2 );
 23   fflush(stdout);
 24 #endif
 25
 26     for( i = 0; i < hmm->num_states; i++ )
 27         for( j = 0; j <= end2; j++ ) Alpha[i][j] = MIN_INT;
 28
 29     rarc = hmm->states[0].rarc;
 30
 31 #if DURATION_CONTROL
 32     if( rarc == NO_ROOP )
 33 #endif
 34         for( i = 0, j = start1; j <= end1; i++, j++ ) Alpha[0][i] = prob[j];
 35 #if DURATION_CONTROL
 36     else
 37     {
 38         index = hmm->transitions[rarc].out_prob_index;
 39         roop_prob = hmm->transitions[rarc].trans_prob;
 40
 41         for( i = 0, j = start1; j <= end1; i++, j++ )
 42         {
 43             if( prob[j] <= MIN_INT ) continue;
 44
 45             ax = prob[j];
 46
 47             for( jj = j+1; jj <= j + dur->durmin[0]; jj++ )
 48                 ax += ( *(hmm->frame_prob[index]+jj-1) + roop_prob );
 49
 50             if( ax <= MIN_INT ) continue;
 51
 52             tx = MIN( end2, j+dur->durmax[0] );
 53             for( jj = j+dur->durmin[0], k = i+dur->durmin[0];
 54                                             jj <= tx; jj++, k++ )
 55             {
 56 #if TRELLIS
 57                 Alpha[0][k] = addlog( Alpha[0][k],
 58                                         ax + dur->durprob[0][k-i] );
 59 #else
 60                 Alpha[0][k] = MAX( Alpha[0][k],
 61                                         ax + dur->durprob[0][k-i] );
 62 #endif
 63                 ax += ( *(hmm->frame_prob[index]+jj) + roop_prob );
 64             }
 65         }
 66     }
 67 #endif
 68
 69
 70 #if DURATION_CONTROL
 71     cal_ialpha_dm( hmm, dur, start1, end2 );
 72 #else
 73     cal_ialpha_m( hmm, dur, start1, end2 );
 74 #endif
 75
 76     final = hmm->num_states - 1;
 77     for( i = start2-start1, j = start2; j <= end2; i++, j++ )
 78         prob[j] = Alpha[final][i];
 79
 80 }
 81
 82
 83 best_iprob( dur, start, end, prob, thre, bestprob, bestframe )
 84 IDUR  *dur;
 85 int   start, end, *bestframe;
 86 int   *prob, thre, *bestprob;
 87 {
 88   int      i, px, haba, haba1, haba2, s0, e0;
 89   double   rint( ), lenx;
 90
 91 #if 0
 92   printf( "start]= %d end]= %d\n", *start, *end );
 93   fflush(stdout);
 94 #endif
 95
 96   if( (start==0) && (end==0) )
 97   {
 98     *bestprob = prob[0];   *bestframe = 0;
 99     if( *bestprob > thre ) return( 1 );
100     else                   return( 0 );
101   }
102
103   *bestprob = MIN_INT;
104   for( i = start; i <= end; i++ )
105   {
106 #if 0
107     printf( "prob[%3d]=%d\n", i, prob[i] ); fflush( stdout );
108 #endif
109     NormProb[i] = (int)rint( (double)prob[i] / (double)i );
110     if( NormProb[i] > *bestprob ) { *bestprob = NormProb[i]; *bestframe = i; }
111   }
112
113   if( *bestprob > thre ) return( 1 );
114   else                   return( 0 );
115
116 }
117
118
119 #if DURATION_CONTROL
120
```

```
LINE #  SOURCE TEXT
  121  cal_ialpha_dm( hmm, dur, start, end )
  122  IMHMM   *hmm;
  123  IDUR    *dur;
  124  int     start, end;
  125  {
  126      register                *stop, *cptr;
  127      register                i, j, jj, k, l, s;
  128      struct transition       *tptr;
  129      int                     out_prob, trans_prob, roop_prob;
  130      int                     ax, rarc, index, tx;
  131
  132      for( s = 1; s < hmm->num_states; s++ )
  133      {
  134          stop = &( hmm->states[s].trans_to[hmm->states[s].num_to] );
  135
  136          for( i = 0, j = start; j <= end; i++, j++ )
  137          {
  138              ax = MIN_INT;
  139              for( cptr = &( hmm->states[s].trans_to[0] ); cptr < stop; cptr++ )
  140              {
  141                  tptr = &(hmm->transitions[*cptr]);
  142
  143                  if( tptr->origin == s ) continue;
  144
  145                  else if( tptr->out_prob_index == NULL_TRANSITION )
  146  #if TRELLIS
  147                      ax = addlog( ax, Alpha[tptr->origin][i]
  148                                       + tptr->trans_prob );
  149  #else
  150                      ax = MAX( ax, Alpha[tptr->origin][i]
  151                                       + tptr->trans_prob );
  152  #endif
  153                  else if( i != 0 )
  154                  {
  155                      out_prob = *(hmm->frame_prob[tptr->out_prob_index] + j-1 );
  156  #if TRELLIS
  157                      ax = addlog( ax, Alpha[tptr->origin][i-1]
  158                                       + tptr->trans_prob + out_prob );
  159  #else
  160                      ax = MAX( ax, Alpha[tptr->origin][i-1]
  161                                       + tptr->trans_prob + out_prob );
  162  #endif
  163                  }
  164              }
  165
  166              if( ax <= MIN_INT ) continue;
  167
  168              rarc = hmm->states[s].rarc;
  169              if( rarc == NO_ROOP )
  170              {
  171                  Alpha[s][i] = ax; continue;
  172              }
  173
  174              index = hmm->transitions[rarc].out_prob_index;
  175              roop_prob = hmm->transitions[rarc].trans_prob;
  176
  177              tx = MIN( end, j+dur->durmin[s] );
  178              for( jj = j+1; jj <= tx; jj++ )
  179                  ax += ( *(hmm->frame_prob[index]+jj-1) + roop_prob );
  180
  181              tx = MIN( end, j+dur->durmax[s] );
  182              for( jj = j+dur->durmin[s], k = i+dur->durmin[s];
  183                                          jj <= tx; jj++, k++ )
  184              {
  185  #if TRELLIS
  186                  Alpha[s][k] = addlog( Alpha[s][k],
  187                                       ax + dur->durprob[s][k-i] );
  188  #else
  189                  Alpha[s][k] = MAX( Alpha[s][k],
  190                                       ax + dur->durprob[s][k-i] );
  191  #endif
  192                  ax += ( *(hmm->frame_prob[index]+jj) + roop_prob );
  193              }
  194          }
  195
  196      }
  197  }
  198
  199
  200  #else
  201
  202  cal_ialpha_m( hmm, dur, start, end )
  203  IMHMM  *hmm;
  204  IDUR   *dur;
  205  int    start, end;
  206  {
  207    register                *stop, *cptr;
  208    register                i, j, s;
  209    struct transition       *tptr;
  210    int                     prev_alphal, out_prob;
  211    int                     q, orig, dest, trans_prob;
  212
  213    for( i = 0, j = start; j <= end; i++, j++ )
  214     {
  215       for( s = 0; s < hmm->num_states; s++ )
  216        {
  217
  218          if( i == 0 && hmm->states[s].is_initial ) continue;
  219
  220          stop = &( hmm->states[s].trans_to[hmm->states[s].num_to] );
  221          for( cptr = &( hmm->states[s].trans_to[0] ); cptr < stop; cptr++ )
  222           {
  223             tptr = &(hmm->transitions[*cptr]);
  224             trans_prob = tptr->trans_prob;
  225             if( tptr->out_prob_index == NULL_TRANSITION )
  226              {
  227                prev_alphal = Alpha[tptr->origin][i];
  228                if( prev_alphal > MIN_INT )
  229  #if TRELLIS
  230                  Alpha[s][i] = addlog((prev_alphal+trans_prob), Alpha[s][i]);
  231  #else
  232                  Alpha[s][i] = MAX((prev_alphal+trans_prob), Alpha[s][i]);
  233  #endif
  234              }
  235             else if( i != 0 )
  236              {
  237                prev_alphal = Alpha[tptr->origin][i-1];
  238                if( prev_alphal > MIN_INT )
  239                 {
  240                   out_prob = *(hmm->frame_prob[tptr->out_prob_index] + j-1 );
```

```
241                     if( out_prob > MIN_INT )
242 #if TRELLIS
243                      Alpha[s][i] = addlog( (prev_alphal+trans_prob+out_prob),
244                                                          Alpha[s][i] );
245 #else
246                      Alpha[s][i] = MAX( (prev_alphal+trans_prob+out_prob),
247                                                          Alpha[s][i] );
248 #endif
249                   }
250                }
251             }
252          }
253       }
254
255 }
256
257 #endif
```

LINE # SOURCE TEXT

```
/*=======================================================================
 *
 *      Initialization --- read HMM models, etc.
 *
 *      Edit by K.Kita & T.Hanazawa.
 *        ATR Interpreting Telephony Research Laboratories.
 *
 *=======================================================================*/

#define     HMM_MAIN

#include <stdio.h>
#include "config.h"
#include "hmmlr.h"
#include "log.h"


/*
 * Phone list file.
 */
#define PHONELIST       "/ifgl/MAU-TH/HMM/list"

/*
 * Logalithm table
 */
#define LOGFILE         "/ifgl/logtbl/log.tbl"

/*=======================================================================*/


#define HASHSIZE        101
#define streq(s,t)      (*(s)==*(t) && strcmp(s,t)==0)

int             *Alpha[MAX_STATES];
static Entry    *EntryTable[HASHSIZE];
static char     *Phonemes[128];
static int      num_model;

init_HMM()
{
        register int    i, nphones = 0;
        char    buf[80], modelfile[80], durfile[MAX_NUM_DUR][80];
        char    phono[17], *malloc();
        Entry   *ep, *lookup();
        FILE    *fp, *fph, *fopen();
        double  floor0 = 1.0e-6, floor[MAX_CODEBOOK];
        double  sigma_model, sigma_state, dur_weight;
        int     num_dur;
        double  durminQ, durmaxQ;

         /*** adaptsion histgram ***/

        char    histfile[80];
        int     num_hist, histsize;
        float   *hist[MAX_CODEBOOK];
        int     *code[MAX_CODEBOOK];

        sigma_model = Sigma_model;
        sigma_state = Sigma_state;
        dur_weight = Dur_weight;

        if( Hmmlist[0] == NULL )  strcpy( Hmmlist, PHONELIST );
        if( Logtable[0] == NULL ) strcpy( Logtable, LOGFILE );

        printf( " Hmm-> %s\n", Hmmlist );
        printf( " ModelDuration-> Sigma: %3.1f\n", sigma_model );
        printf( " StateDuration-> Sigma: %3.1f (*%3.1f) Weight: %5.2f\n\n",
                                        sigma_state, Nsd, dur_weight );
        if( Histlist[0] != NULL )
                printf( " Histlist-> %s  Top: %d\n", Histlist, Tophist );
        fflush(stdout);

        /*
         * Speaker adaptation histgram
         */
        if( Histlist[0] != NULL )
        {
            fph = fopen( Histlist, "r" );
            if( fph == NULL )
            {
                fprintf( stderr, "can't open %s\n", Histlist );
                printf( "can't open %s\n", Histlist );
                exit(1);
            }

            fscanf( fph, "%d", &num_hist );
            for( i = 0; i < num_hist; i++ )
            {
                fscanf( fph, "%s %d", histfile, &histsize );
                printf( " %s %d\n", histfile, histsize ); fflush(stdout);
                get_hist( histfile, Tophist, histsize, &hist[i], &code[i] );
            }
            fclose(fph);
            printf( "\n" ); fflush(stdout);
        }

        /*
         * initialize EntryTable
         */
        for( i = 0; i < HASHSIZE; i++ ) EntryTable[i] = NULL;

        /*
         * Allocate fuzzy-data area.
         */
        for(i = 0; i < MAX_CODEBOOK; i++)
        {

            Fdata[i].code =
                  (unsigned char*)malloc( MAXFRAME * MAX_FUZZY
                                                * sizeof(unsigned char) );
            if( Fdata[i].code == NULL )
                { printf( "can't allocate fuzzy-code\n" ); exit(1); }

            Fdata[i].m = (float*)malloc( MAXFRAME * MAX_FUZZY
                                               * sizeof(float) );
            if( Fdata[i].m== NULL )
                { printf( "can't allocate fuzzy-m\n" ); exit(1); }

        }
```

```
LINE #  SOURCE TEXT
121
122         /*
123          * Allocate working area.
124          */
125         alloc_ibuf( MAXFRAME+1, MAX_STATES );
126
127         /*
128          * Read log-tabel
129          */
130         read_logtbl( Logtable );
131
132         /*
133          * Read phone model & duration.
134          */
135         for(i = 0; i < MAX_CODEBOOK; i++) floor[i] = floor0;
136
137         if((fp = fopen( Hmmlist , "r")) == NULL)
138                 fatal_error("Cannot open %s.\n", Hmmlist );
139
140         num_model = 0;
141         while(1)
142         {
143                 if(fgets(buf, sizeof(buf), fp) == NULL)
144                         break;
145                 if(strlen(buf) == 1 || buf[0] == '#')
146                         continue;
147                 if( sscanf(buf, "%s %s %d", modelfile, phone, &num_dur) != 3 )
148                         fatal_error("Error in %s.\n", Hmmlist );
149                 for( i = 0; i < num_dur; i++ ) fscanf(fp, "%s", durfile[i] );
150
151                 /*
152                  * Check existence of phone
153                  */
154                 if(lookup(phone) != NULL)
155                 {
156                 error("Phone '%s' already exists. Skip loading...\n", phone);
157                         continue;
158                 }
159
160                 /*
161                  * Prepare entry structure.
162                  */
163                 if((ep = (Entry*)malloc(sizeof(Entry))) == NULL)
164                         goto NoMemory;
165                 if((ep->phone = (char*)malloc(strlen(phone)+1)) == NULL)
166                         goto NoMemory;
167                 strcpy(ep->phone, phone);
168                 if((ep->model = (IMHMM*)malloc(sizeof(IMHMM))) == NULL)
169                         goto NoMemory;
170                 for( i = 0; i < num_dur; i++ )
171                     if((ep->duration[i] = (IDUR*)malloc(sizeof(IDUR))) == NULL)
172                             goto NoMemory;
173
174                 /*
175                  * Read model.
176                  */
177                 /*fprintf(stderr, "-> Loading %s ...", modelfile );*/
178                 fprintf(stderr, "-> Loading %s\n", modelfile );
179                 fflush(stderr);
180                 read_imhmm(modelfile, floor, ep->model ); num_model++;
181
182                 /*
183                  * frame prob buffer allocate.
184                  */
185                 for( i = 0; i < ep->model->num_omatrix; i++ )
186                 {
187                     ep->model->frame_prob[i]
188                         = (int*)malloc( MAXFRAME * sizeof(int) );
189                     if( ep->model->frame_prob[i] == NULL )
190                     {
191                         fprintf( stderr, "can't alloctate frame_prob\n" );
192                         printf( "can't alloctate frame_prob\n" );
193                         exit(1);
194                     }
195                 }
196
197                 /*
198                  * Model adaptation.
199                  */
200                 /* print_mhmm( ep->model ); */
201                 if( Histlist[0] != NULL )
202                 {
203                     if( num_hist != ep->model->num_book )
204                     {
205                         printf( "num_hist(%d) != hmm(%d)\n",
206                                 num_hist, ep->model->num_book );
207                         exit(1);
208                     }
209                     for( i = 0; i < num_hist; i++ )
210                         adapt_hmm( ep->model, i, floor[i],
211                                    Tophist, hist[i], code[i] );
212                     /* print_mhmm( ep->model ); */
213                 }
214
215                 /*fprintf(stderr, "Ok\n");*/
216
217                 /*
218                  * Read duration.
219                  */
220                 for( i = 0; i < num_dur; i++ )
221                 {
222                     /*fprintf(stderr, "-> Loading %s...", durfile[i] );*/
223                     fprintf(stderr, "-> Loading %s\n", durfile[i] );
224                     fflush(stderr);
225                     read_idur( durfile[i], sigma_model,
226                                            sigma_state, ep->duration[i] );
227                     pow_idur( ep->duration[i], dur_weight );
228                     /*fprintf(stderr, "Ok\n");*/
229                 }
230                 ep->num_dur = num_dur;
231
232                 /*
233                  * Install entry structure.
234                  */
235                 install(ep);
236                 Phonemes[nphones] = ep->phone;
237                 ++nphones;
238         }
239         Phonemes[nphones] = NULL;
240         fclose(fp);
```

LINE # SOURCE TEXT

```
241
242         if( Histlist[0] != NULL )
243         {
244             for( i = 0; i < num_hist; i++ )
245             {
246                 free( hist[i] ); free( code[i] );
247             }
248         }
249
250         return(nphones);
251   NoMemory:
252         fatal_error("Cannot allocate memory for phone '%s'\n", phone);
253
254 }
255
256
257 unsigned        hash(phone)
258 char    *phone;
259 {
260         register unsigned       val;
261
262         for( val= 0; *phone != NULL; )
263                 val += (unsigned)*phone++;
264         return( val%HASHSIZE );
265 }
266
267
268
269 Entry   *lookup(phone)
270 char    *phone;
271 {
272         register Entry  *p;
273         unsigned        hash();
274
275         for(p = EntryTable[hash(phone)]; p != NULL; p = p->next)
276         {
277                 if(strcmp(phone, p->phone) == 0)
278                         return(p);
279         }
280         return(NULL);
281 }
282
283
284 install(ep)
285 Entry   *ep;
286 {
287         Entry   *ep2, *lookup();
288         unsigned        val, hash();
289         char    *malloc();
290
291         if((ep2 = lookup(ep->phone)) != NULL)
292         {
293                 error("Already exists entry '%s'\n", ep->phone);
294                 return(FALSE);
295         }
296         val = hash(ep->phone);
297         ep->next = EntryTable[val];
298         EntryTable[val] = ep;
299         return(TRUE);
300 }
301
302
303 CheckPhoneme(s)
304 char    *s;
305 {
306         register int    i;
307         int     ok = 0;
308
309         for(i = 0; Phonemes[i] != NULL; i++)
310         {
311                 if(strcmp(s, Phonemes[i]) == 0)
312                 {
313                         ++ok;
314                         break;
315                 }
316         }
317         if(!ok)
318         {
319                 if(streq(s, "ch") || streq(s, "ts") || streq(s, "p")
320                 || streq(s, "t") || streq(s, "k") || streq(s, "b")
321                 || streq(s, "d") || streq(s, "g") || streq(s, "gy"))
322                         return(TRUE);
323                 return(FALSE);
324         }
325         return(TRUE);
326 }
327
328
329 Read_Fuzzy( fp )        /* minor modification by kita, 07-Mar-1990 */
330 FILE  *fp;
331 {
332         int     st, id, fd, start, end, size, i, j, k, lx, num_dat, c;
333         char    input[120];
334         char    buf[BUFSIZ], iname[80], datafile[MAX_CODEBOOK][80];
335         double  sp, ep, sp2, ep2, period;
336
337         if(fgets(buf, sizeof(buf), fp) == NULL)
338                 return(EOF);
339
340         if(buf[0] == '#')
341                 return(FALSE);
342
343         sscanf( buf, "%d %s %d %s %lf %lf",
344                                 &id, iname, &num_dat, input, &sp, &ep );
345
346         for( i = 0; i < num_dat; i++ )
347         {
348             if( fscanf( fp, "%s\n", datafile[i] ) != 1 )
349             {
350                 printf( "Error in labelfile\n" ); exit(1);
351             }
352         }
353
354         sp2 = sp - Add_Time;
355         ep2 = ep + Add_Time;
356         period = Frame_Period / (double)Mabiki;
357
358         start = (int)( sp2 / period );
359         end =   (int)( ep2 / period );
360
```

LINE # SOURCE TEXT

```
361         /*
362          * Read data-file.
363          */
364         for( i = 0; i < num_dat; i++ )
365                 read_fdata( datafile[i], start, end, &Fdata[i] );
366
367         N_frames = Fdata[0].length;
368         printf("(%03d) %s  %7.1f %7.1f   |%s|  %d frames\n", id, iname, sp2, ep2, input, N_frames);
369         fflush(stdout);
370
371 /*****
372         if((c = getc(fp)) == '!')
373         {
374                 while(getc(fp) != '\n')
375                         ;
376                 return(FALSE);
377         }
378         else
379                 ungetc(c, fp);
380 *****/
381
382         return(TRUE);
383 }
384
385
386 read_fdata( fn, bgn, end, fdata )
387 char   *fn;
388 int    bgn, end;
389 FDATA  *fdata;
390 {
391   int     i, j, ptr, rb, rbi, rbf, size_1, size_c, size_m, fd, leng;
392   long    offset0, offset1;
393
394   fd = open( fn, 0 );
395   if( fd == -1 ) { printf( "can't open %s\n", fn ); exit(1); }
396
397   rbi = sizeof(int); rbf = sizeof(float);
398
399   read( fd, &(fdata->fuzziness), rbf );
400   read( fd, &(fdata->num_fuzzy), rbi );
401   read( fd, &(fdata->length),    rbi );
402
403 #if DEC_TO_IEEE
404   float_dec_to_ieee( &(fdata->fuzziness), 1 );
405   long_dec_to_ieee( &(fdata->num_fuzzy), 1 );
406   long_dec_to_ieee( &(fdata->length), 1 );
407 #endif
408
409 #if 0
410   printf( " fuzziness--> %3.1f\n", fdata->fuzziness );
411   printf( " num_fuzzy--> %d\n",    fdata->num_fuzzy );
412   printf( " length-----> %d\n",    fdata->length );
413   printf( " bgn(%d) end(%d)\n", bgn, end );
414 #endif
415
416   size_1 = fdata->num_fuzzy * ( sizeof(unsignod char) + sizeof(float) );
417   offset0 = (long)( size_1 * bgn );
418   if( lseek( fd, offset0, 1 ) < 0 ) { printf( "can't lseek0\n" );  exit(1); }
419
420   size_c = fdata->num_fuzzy * sizeof(unsigned char);
421   size_m = fdata->num_fuzzy * sizeof(float);
422
423   ptr = 0; leng = 0;
424   offset1 = (long)( size_1 * (Mabiki-1) );
425   for( i = bgn; i <= end; i+= Mabiki )
426    {
427     rb = read( fd, (fdata->code)+ptr, size_c );
428     if( rb != size_c )
429       { printf( "error1: rb(%d) != size_c(%d)\n", rb, size_c ); exit(1); }
430
431     rb = read( fd, (fdata->m)+ptr, size_m );
432     if( rb != size_m )
433       { printf( "error2: rb(%d) != size_c(%d)\n", rb, size_m ); exit(1); }
434
435 #if DEC_TO_IEEE
436     float_dec_to_ieee( (fdata->m)+ptr, fdata->num_fuzzy );
437 #endif
438
439     ptr += fdata->num_fuzzy; leng++;
440
441     if( lseek( fd, offset1, 1 ) < 0 ) { printf( "can't lseek1\n" ); exit(1); }
442
443    }
444   close( fd );
445
446   fdata->length = leng;
447
448 }
449
450
451 alloc_ibuf( longest, states )
452 int  longest, states;
453 {
454   int   i;
455   char  *malloc( );
456
457   for( i = 0; i < states; i++ )
458    {
459       Alpha[i] = (int*)malloc( (longest+1) * sizeof(int) );
460       if( Alpha[i] == NULL ) { printf( "can't allocate Alpha\n" ); exit(1); }
461    }
462 }
463
464
465 cal_frame_prob( )
466 {
467     register Entry      *ep;
468     register            i, num;
469
470     num = 0;
471     for( i = 0; i < HASHSIZE; i++ )
472     {
473         for( ep = EntryTable[i]; ep != NULL; ep = ep->next )
474         {
475             /* printf( "HMM(%s)\n", ep->phone ); */
476             cal_frame_prob0( ep->model );
477             num++;
478         }
479     }
480
```

```
LINE #  SOURCE TEXT
481  #if 0
482      printf( "num_model= %d  num= %d\n", num_model, num );
483  #endif
484
485      if( num != num_model )
486      {
487          printf( "AHO !\n" );
488          fprintf( stderr, "AHO !\n" );
489          exit(1);
490      }
491  }
492
493
494  cal_frame_prob0( hmm )
495  IMHMM  *hmm;
496  {
497      register    i, j;
498
499      for( i = 0; i < hmm->num_omatrix; i++ )
500          for( j = 0; j < Fdata[0].length; j++ )
501              *( hmm->frame_prob[i] + j )= cal_fuzzy( hmm, i, j );
502  }
503
504
505  #define  DEBUG      0
506
507  cal_fuzzy( hmm, index, time )
508  IMHMM  *hmm;
509  int    index, time;
510  {
511    register      i, j;
512    int           ptr, code, outp, tx, px_int;
513    float         px, m;
514    double        log( ), rint( );
515
516    outp = 0;
517
518    for( i = 0; i < hmm->num_book; i++ )
519     {
520       tx = Fdata[i].num_fuzzy * time;
521       code =  (int)*( Fdata[i].code + tx );
522       m =  *( Fdata[i].m + tx );
523
524  #if DEBUG
525       printf( "code[t=%3d][book=%d][%d]= %03d  m= %6.4f\n",
526                                                time, i+1, 1, code, m );
527  #endif
528
529       ptr = hmm->num_vector[i] * index + code;
530
531       px =  hmm->output_prob[i][ptr] * m;
532
533       for( j = 1; j < Fdata[i].num_fuzzy; j++ )
534        {
535          code =  (int)*( Fdata[i].code + tx + j );
536          m =  *( Fdata[i].m + tx + j );
537  #if DEBUG
538          printf( "code[t=%3d][book=%d][%d]= %03d  m= %6.4f\n",
539                                              time, i+1, j+1, code, m );
540  #endif
541          ptr = hmm->num_vector[i] * index + code;
542
543          px += hmm->output_prob[i][ptr] * m;
544        }
545       if( px != 0.0 )
546          px_int = (int)rint( logx(Lgtbl.b,(double)px) );
547       else
548          px_int = MIN_INT;
549
550       outp += px_int;
551     }
552
553  #if DEBUG
554    printf( "outp= %d\n", outp ); fflush(stdout);
555  #endif
556
557    if( outp < MIN_INT ) outp = MIN_INT;
558
559    return( outp );
560  }
```

LINE # | SOURCE TEXT

```
 1  #define     LOG_MAIN
 2
 3  #include <stdio.h>
 4  #include "config.h"
 5  #include "log.h"
 6
 7  read_logtbl( fname )
 8  char   *fname;
 9  {
10    int  fd, rbl;
11
12    fd = open( fname, 0 );
13    if( fd == -1 ) { printf( "can't open %s\n", fname ); exit(1); }
14
15    rbl = read( fd, &(Lgtbl.b), sizeof(double) );
16    rbl = read( fd, &(Lgtbl.size), sizeof(int) );
17
18  #if DEC_TO_IEEE
19    double_d_dec_to_ieee(&(Lgtbl.b), 1);
20    long_dec_to_ieee(&(Lgtbl.size), 1);
21  #endif
22
23    /*printf( "\nb=%f  size=%d\n\n", Lgtbl.b, Lgtbl.size );*/
24
25    Lgtbl.table = (int*)malloc( Lgtbl.size * sizeof(int) );
26    if( Lgtbl.table == NULL ){ printf( "can't allocate logtable\n" ); exit(1); }
27
28    rbl = read( fd, Lgtbl.table, (Lgtbl.size)*sizeof(int) );
29    if( rbl != (Lgtbl.size)*sizeof(int) )
30     {
31       printf( "can't read %s ( read(%d) != size(%d) )\n",
32                                  fname, rbl/sizeof(int), Lgtbl.size );
33       exit(1);
34     }
35
36  #if DEC_TO_IEEE
37    long_dec_to_ieee(Lgtbl.table, Lgtbl.size);
38  #endif
39
40    close(fd );
41
42  }
43
44
45  addlog( x, y )
46  int    x, y;
47  {
48    int  n;
49
50  #if 0
51    printf( "x= %d\n", x );
52    printf( "y= %d\n", y );
53    fflush(stdout);
54  #endif
55
56    if( x > y )
57     {
58       n = x - y;
59  #if 0
60       printf( "n=%d\n", n );
61       fflush(stdout);
62  #endif
63       if( n >= Lgtbl.size ) return( x );
64       return( x + Lgtbl.table[n] );
65     }
66    else
67     {
68       n = y - x;
69       /* printf( "n=%d\n", n ); */
70       if( n >= Lgtbl.size ) return( y );
71       return( y + Lgtbl.table[n] );
72     }
73  }
```

LINE # SOURCE TEXT

```
#include "imhmm.h"
#include  "log.h"

read_imhmm1( file, floor, phm )
char    *file;
double  floor[ ];
IMHMM   *phm;
{
  FILE      *fp, *fopen( );
  int       num, mat_num, from, to, disp;
  int       nmat, nstat, ninit, nfin, narc;
  double    log( ), rint( ), outp[MAX_ALPHABET], trans_prob, x;
  int       nbook, csize[MAX_CODEBOOK];
  char      *malloc( ), dummy[77];
  register  *stop, *cptr;
  register  i, j, k, s;
  struct transition   *tptr;

  fp = fopen( file, "r" );
  if( fp == NULL ) { printf( "can't open %s\n", file ); exit(1); }

  fscanf( fp, "%s %d", dummy, &nbook );
  if( nbook > MAX_CODEBOOK )
   {
     printf( "nbook(%d) > MAX_CODEBOOK(%d)\n", nbook, MAX_CODEBOOK );
     exit(1);
   }
  fscanf( fp, "%s %d", dummy, &nmat );

  for( i = 0; i < nbook; i++ )
   {
     fscanf( fp, "%s %d", dummy, &csize[i] );
     /* printf( "nmat= %d  csize= %d\n", nmat, csize[i] ); */
     phm->output_prob[i] = (float*)malloc(sizeof(float) * nmat * csize[i]);
     if( phm->output_prob[i] == NULL )
      { printf( "can't allocate hmm outprob\n" ); exit(1); }
     for( j = 0; j < nmat; j++ )
      {
        disp = j * csize[i]; x = 0.0;
        for( k = 0; k < csize[i]; k++ )
         {
           fscanf( fp, "%le", &outp[k] );
           if( outp[k] < floor[i] ) outp[k] = floor[i];
           x += outp[k];
         }
        for( k = 0; k < csize[i]; k++ )
         *(phm->output_prob[i]+disp+k) = (float)( outp[k] / x );
      }
   }

  fscanf( fp, "%d", &nstat );
  if( nstat > MAX_STATES )
   { printf( "states(%d) > MAX_STATES(%d)\n", nstat, MAX_STATES ); exit(1); }
  phm->states = (struct state *)malloc( sizeof(struct state) * nstat );
  if( phm->states == NULL )
    { printf( "can't allocate hmm_states\n" ); exit(1);}
  for( i = 0; i < nstat; i++ )
   {
     phm->states[i].label = i;
     phm->states[i].num_from = 0;
     phm->states[i].num_to = 0;
     phm->states[i].is_initial = phm->states[i].is_final = 0;
   }

  fscanf( fp, "%d", &ninit );
  for( i = 0; i < ninit; i++ )
   {
     fscanf( fp, "%d", &num );
     phm->states[num].is_initial = 1;
   }

  fscanf( fp, "%d", &nfin );
  for( i = 0; i < nfin; i++ )
   {
     fscanf( fp, "%d", &num );
     phm->states[num].is_final = 1;
   }

  fscanf( fp, "%d", &narc );
  if( narc > MAX_ARCS )
   { printf( "arcs(%d) > MAX_ARCS(%d)\n", narc, MAX_ARCS ); exit(1); }
  phm->transitions =
    (struct transition *)malloc( sizeof (struct transition) * narc );
  if( phm->transitions == NULL )
    { printf( "can't allocate hmm_transitions\n" ); exit(1);}
  for( i = 0; i < narc; i++ )
   {
     fscanf( fp, "%d %d %le %d", &from, &to, &trans_prob, &mat_num );

     if( trans_prob != 0.0 )
        phm->transitions[i].trans_prob = (int)rint( logx(Lgtbl.b,trans_prob) );
     else
        phm->transitions[i].trans_prob = MIN_INT;

     phm->transitions[i].origin = from;
     phm->transitions[i].destination = to;
     phm->transitions[i].out_prob_index = mat_num;

     phm->states[from].trans_from[phm->states[from].num_from++] = i;
     phm->states[to].trans_to[phm->states[to].num_to++] = i;
   }

  fclose( fp );

  phm->num_omatrix  = nmat;
  phm->num_states   = nstat;
  phm->num_initial  = ninit;
  phm->num_final    = nfin;
  phm->num_arcs     = narc;
  phm->num_book     = nbook;

  for( i = 0; i < nbook; i++ ) phm->num_vector[i]  = csize[i];

  for( s = 0; s < phm->num_states; s++ )
   {
     phm->states[s].rarc = NO_ROOP;
     stop = &( phm->states[s].trans_to[phm->states[s].num_to] );
     for( cptr = &( phm->states[s].trans_to[0] ); cptr < stop; cptr++ )
      {
        tptr = &(phm->transitions[*cptr]);
```

```
LINE #  SOURCE TEXT
  121       if( (tptr->origin == s) && (tptr->trans_prob != MIN_INT) )
  122          { phm->states[s].rarc = *cptr; break; }
  123       }
  124    }
  125 }
```

LINE # SOURCE TEXT

```
  1 #include "imhmm.h"
  2 #include "idur.h"
  3 #include "log.h"
  4
  5 #define  MIN_HABA       1.0
  6 #define  MIN_SD         0.15
  7
  8 extern double   Nsd;
  9
 10 read_idur( dur_file, sigma_model, sigma_state, dur )
 11 char    *dur_file;
 12 double  sigma_model, sigma_state;
 13 IDUR    *dur;
 14 {
 15   int     i, j, max, ptr, dummy, st, num, devoc_flg;
 16   char    buf[87];
 17   FILE    *fp, *fopen( );
 18   double  rint( ), exp( ), log( ), mean, sd, var, x, haba;
 19   char    *malloc( );
 20
 21   fp = fopen( dur_file, "r" );
 22   if( fp == NULL ) { printf( "can't open %s\n", dur_file ); exit(1); }
 23
 24   while( 1 )
 25    {
 26      st = read_linef( fp, buf, 87 );
 27      if( st == EOF ) { printf( "can't read %s\n", dur_file ); exit(1); }
 28      if( !strcmp( buf,"&" ) ) break;
 29    }
 30
 31   st = read_linef( fp, buf, 87 );
 32   sscanf( buf, "%d %d %le %le",
 33              &(dur->lenmin), &(dur->lenmax), &(dur->lenav), &(dur->lensd) );
 34
 35   if( dur->lensd != FLAT )
 36    {
 37      if( dur->lensd < MIN_SD ) dur->lensd = MIN_SD;
 38
 39      dur->lenmin2 = (int)rint( dur->lenav - sigma_model*(dur->lensd) );
 40      if( dur->lenmin2 < 0 ) dur->lenmin2 = 0;
 41      dur->lenmax2 = (int)rint( dur->lenav + sigma_model*(dur->lensd) );
 42
 43      dur->lenprob= (int*)malloc( (dur->lenmax2+1) * sizeof(int) );
 44      if( dur->lenprob == NULL )
 45          { printf( "can't allocate lenprob\n" ); exit(1); }
 46
 47      mean = dur->lenav;
 48      var = (dur->lensd) * (dur->lensd);
 49
 50      for( i = 0; i <= dur->lenmax2; i++ )
 51       {
 52         x = (-1.0) * ((double)i-mean) * ((double)i-mean) / ( 2.0 * var );
 53         x = exp( x );
 54         if( x > 0.0 )
 55           dur->lenprob[i] = (int)rint( logx( Lgtbl.b, x ) );
 56         else
 57           dur->lenprob[i] = MIN_INT;
 58       }
 59    }
 60   else
 61    {
 62      dur->lenmin2 = dur->lenmin;
 63      dur->lenmax2 = dur->lenmax;
 64
 65      dur->lenprob= (int*)malloc( (dur->lenmax2+1) * sizeof(int) );
 66      if( dur->lenprob == NULL )
 67          { printf( "can't allocate lenprob\n" ); exit(1); }
 68
 69      for( i = 0; i < dur->lenmin2; i++ )             dur->lenprob[i] = MIN_INT;
 70      for( i = dur->lenmin2; i <= dur->lenmax2; i++ ) dur->lenprob[i] = 0;
 71    }
 72
 73   st = read_linef( fp, buf, 87 );
 74   sscanf( buf, "%d", &(dur->states) );
 75
 76   for( i = 0; i < dur->states; i++ )
 77    {
 78      st = read_linef( fp, buf, 87 );
 79      devoc_flg = 0;
 80      sscanf( buf, "%d %le %le %d", &num, &mean, &sd, &devoc_flg );
 81
 82      sd *= Nsd;
 83      if( sd < MIN_SD ) sd = MIN_SD;
 84
 85      if( dur->lensd != FLAT )
 86       {
 87         haba = sigma_state * sd;
 88         if( haba < MIN_HABA ) haba = MIN_HABA;
 89         dur->durmin[i] = (int)rint( mean - haba );
 90         if( dur->durmin[i] < 0 ) dur->durmin[i] = 0;
 91         dur->durmax[i] = (int)rint( mean + haba );
 92       }
 93      else
 94       {
 95         dur->durmin[i] = dur->lenmin - 1;
 96         dur->durmax[i] = dur->lenmax - 1;
 97       }
 98
 99      dur->durprob[i] = (int*)malloc( (dur->durmax[i]+1) * sizeof(int) );
100      if( dur->durprob[i] == NULL )
101          { printf( "can't allocate dur\n" ); exit(1); }
102
103      var = sd * sd;
104      for( j = 0; j <= dur->durmax[i]; j++ )
105       {
106         if( dur->lensd != FLAT )
107          {
108             x = (-1.0) * ((double)j-mean) * ((double)j-mean) / ( 2.0 * var );
109             x = exp( x );
110             if( x > 0.0 )
111               dur->durprob[i][j] = (int)rint( logx( Lgtbl.b, x ) );
112             else
113               dur->durprob[i][j] = MIN_INT;
114          }
115         else  dur->durprob[i][j] = 0;
116 #if 0
117         printf( "x=%e  log(x)=%d\n", x, dur->durprob[i][j] );
118 #endif
119       }
120
```

LINE # | SOURCE TEXT

```
121      if( devoc_flg ) /*** devocalization ***/
122       {
123  /*
124         printf( "DEVOC: %s\n", dur_file ); fflush(stdout);
125  */
126         dur->lenmin = 0;
127         dur->durmin[i] = 0;
128         dur->durprob[i][0] = 0;
129       }
130
131     }
132
133   fclose( fp );
134  }
135
136
137  pow_idur( dur, weight )
138  IDUR    *dur;
139  double  weight;
140  {
141    int     i, j, ptr;
142    double  pow( ), rint( );
143
144    for( i = 0; i <= dur->lenmax2; i++ )
145       dur->lenprob[i] = (int)rint( (double)(dur->lenprob[i]) * weight );
146
147    for( i = 0; i < dur->states; i++ )
148     {
149       if( dur->durmax[i] == 0 ) continue;
150       for( j = 0; j <= dur->durmax[i]; j++ )
151        dur->durprob[i][j] = (int)rint( (double)(dur->durprob[i][j]) * weight );
152     }
153
154  }
155
156
157  print_idur( dur )
158  IDUR  *dur;
159  {
160    int  s, j;
161
162    printf( "&\n" );
163    printf( "%3d  %3d  %e  %e\n", dur->lenmin, dur->lenmax,
164                                  dur->lenav, dur->lensd );
165
166    printf( "%3d\n", dur->states );
167    fflush(stdout);
168    for( s = 0; s < dur->states; s++ )
169     {
170       printf( "%3d\n", s );
171       for( j = dur->durmin[s]; j <= dur->durmax[s]; j++ )
172        {
173          printf( "%3d  %11d\n", j, dur->durprob[s][j] );
174          fflush(stdout);
175        }
176     }
177    printf( "&\n" );
178    fflush(stdout);
179  }
180
181
182  read_linef( fp, buf, size )
183  FILE  *fp;
184  char  *buf;
185  int   size;
186  {
187    int   i, k;
188    char  c;
189
190    for( i = 0; i < size; i++ ) buf[i] = 0x00;
191
192    k = 0;
193    for( i = 0; ; i++ )
194     {
195       c = (char)getc( fp );
196       if( c == (char)EOF  ) return( (int)EOF );
197       if( c == 0x0a       ) break;
198       k++;
199       if( k > size ) return( -2 );
200       buf[i] = c;
201     }
202  /*
203    printf( "%s \n", buf );
204  */
205    return( k );
206  }
```

LINE # SOURCE TEXT

```
  1 /*****************************************************/
  2 /*                                                   */
  3 /*      Data Format Exchange Program                 */
  4 /*                                                   */
  5 /*      Doc format to Ieee format                    */
  6 /*                                                   */
  7 /*      Warnning  This program does not              */
  8 /*                consider about over/under flow.    */
  9 /*                                                   */
 10 /*                 Autor         : I Fukuda/APOLLO   */
 11 /*                 Date          : 5-JUNE-'87        */
 12 /*****************************************************/
 13
 14 /*****************************************************/
 15 /*                                                   */
 16 /*      Usage                                        */
 17 /*                                                   */
 18 /*      % dtoi -[fdgsl] <file >file1                 */
 19 /*      % dtoi -[fdgsl] file [file file file ...]    */
 20 /*                                                   */
 21 /*        -f    32 bit foating number data           */
 22 /*        -d    64 bit ieee floating to dec d float  */
 23 /*        -g    64 bit ieee floating to dec g float  */
 24 /*        -s    16 bit short integer                 */
 25 /*        -l    32 bit long integer                  */
 26 /*                                                   */
 27 /*                                                   */
 28 /*****************************************************/
 29 #include <stdio.h>
 30 #include <sys/file.h>
 31 #include <signal.h>
 32
 33 #if 0
 34
 35 #define    MAX_D_BUF 16384
 36 char       d_buffer[MAX_D_BUF];
 37
 38 main(argc,argv)
 39 int      argc;
 40 char     *argv[];
 41 {
 42         int   fpoi;
 43         int     get_sigfpe();
 44       signal(SIGFPE,get_sigfpe);
 45         if (argc <= 1) usage();
 46         if (*argv[1] != '-') usage();
 47         switch( *( argv[1] + 1 )) {
 48         case 'd':
 49                 break;
 50         case 'g':
 51                 break;
 52         case 'f':
 53                 break;
 54         case 's':
 55                 break;
 56         case 'l':
 57                 break;
 58         default :
 59                 usage();
 60                 break;
 61         }
 62         if ( argc == 2 ) cnv_stdinout(*(argv[1]+1));
 63         if ( argc > 2 )  {
 64                 for ( fpoi = 2 ; fpoi < argc ; fpoi++) {
 65                         cnv_file(*(argv[1]+1),argv[fpoi]);
 66                 }
 67         }
 68 }
 69
 70 get_sigfpe(sig,code,scp,srp)
 71 int         sig,code;
 72 struct      sigcontext *scp;
 73 struct      sigregs    *srp;
 74 {
 75 fprintf(stderr,"Signal SIGFPE code=%d!!\n",code);
 76 exit(-1);
 77 }
 78
 79 cnv_stdinout(type)
 80 char        type;
 81 {
 82         int count;      /* read char count */
 83         while((count = read(0, d_buffer, MAX_D_BUF)) > 0 ) {
 84                 switch( type ) {
 85                 case 'd':
 86                         double_d_dec_to_ieee(d_buffer,count/8);
 87                         break;
 88                 case 'g':
 89                         double_g_dec_to_ieee(d_buffer,count/8);
 90                         break;
 91                 case 'f':
 92                         float_dec_to_ieee(d_buffer,count/4);
 93                         break;
 94                 case 's':
 95                         short_dec_to_ieee(d_buffer,count/2);
 96                         break;
 97                 case 'l':
 98                         long_dec_to_ieee(d_buffer,count/4);
 99                         break;
100                 default :
101                         usage();
102                         break;
103                 }
104                 write(1,d_buffer,count);
105         }
106 }
107 cnv_file(type,file_name)
108 char        type,*file_name;
109 {
110         int   fd,pos,count;
111         if ((fd=open(file_name,O_RDWR)) >= 0) {
112                 while ((count = read(fd, d_buffer, MAX_D_BUF)) > 0 ) {
113                         switch( type ) {
114                         case 'd':
115                                 double_d_dec_to_ieee(d_buffer,count/8);
116                                 break;
117                         case 'g':
118                                 double_g_dec_to_ieee(d_buffer,count/8);
119                                 break;
120                         case 'f':
```

```
LINE #  SOURCE TEXT
121                                 float_dec_to_ieee(d_buffer,count/4);
122                                 break;
123                         case 's':
124                                 short_dec_to_ieee(d_buffer,count/2);
125                                 break;
126                         case 'l':
127                                 long_dec_to_ieee(d_buffer,count/4);
128                                 break;
129                         default :
130                                 usage();
131                                 break;
132                         }
133                         pos=lseek(fd,-count,L_INCR);
134                         write(fd,d_buffer,count);
135                 }
136                 close(fd);
137         }
138         else perror(file_name);
139 }
140
141 usage()
142 {
143         fprintf(stderr,"\n Usage of DEC to IEEE format converter\n");
144         fprintf(stderr,"     dtoi -[dgfls] file [file file file]\n");
145         fprintf(stderr,"     dtoi -[dgfls] <file >file\n");
146         exit();
147 }
148
149 #endif
150
151
152
153 /*************************************************/
154 /*                                               */
155 /*         This function convert                 */
156 /*   DEC D FLOAT to IEEE double precision float  */
157 /*                                               */
158 /*************************************************/
159
160 double_d_dec_to_ieee(buffer,count)
161
162 union all_mighty {
163         double          fval;
164         unsigned int    ival[2];
165         char            cv[8];
166 }
167 *buffer;
168 int       count;
169 {
170         union     all_mighty  buf;
171         char      work;
172         int       cnt,cnt1;
173         int       iwork[2],jwork,jwork1;
174         for( cnt = 0 ; cnt < count ; cnt++ ) {
175                 buf.ival[0] = buffer[cnt].ival[0];
176                 buf.ival[1] = buffer[cnt].ival[1];
177
178                 /*********************** swap byte ***********************/
179                 work = buf.cv[0];
180                 buf.cv[0] = buf.cv[1];
181                 buf.cv[1] = work;
182                 work = buf.cv[2];
183                 buf.cv[2] = buf.cv[3];
184                 buf.cv[3] = work;
185                 work = buf.cv[4];
186                 buf.cv[4] = buf.cv[5];
187                 buf.cv[5] = work;
188                 work = buf.cv[6];
189                 buf.cv[6] = buf.cv[7];
190                 buf.cv[7] = work;
191
192
193                 /************************ shift right 3 **********************/
194                 buf.ival[0] = ( buf.ival[0] & 0x7fffffff );
195                 for( cnt1 = 0 ; cnt1 < 3 ; cnt1++ ) {
196                         jwork = ( buf.ival[0] & 0x00000001 );
197                         buf.ival[0]>>=1;
198                         jwork1 = ( buf.ival[1] & 0x00000001 );
199                         buf.ival[1]>>=1;
200                         if ( jwork > 0 ) buf.ival[1] |= 0x80000000;
201                 }
202                 ;
203                 if (jwork1 > 0) buf.ival[1] += 0x00000001;
204                 /************************ set exp ************************/
205                 buffer[cnt].ival[0] = ((( buf.ival[0] + 0x37e00000 ) & 0x7fffffff) | ((buffer[cnt].ival[0] & 0x800000)<<8 ));
206                 buffer[cnt].ival[1] = buf.ival[1];
207         }
208 }
209
210
211
212 /*************************************************/
213 /*                                               */
214 /*                                               */
215 /*         This function convert                 */
216 /*         DEC FLOAT to IEEE float               */
217 /*                                               */
218 /*************************************************/
219
220 float_dec_to_ieee(buffer,count)
221 union all_myghty {
222         float  fval;
223         int    ival;
224         char   cv[4];
225 }
226 *buffer;
227
228 int               count;
229 {
230         char      work;
231         int       cnt;
232
233         /*********************** swap byte ***********************/
234         for( cnt = 0 ; cnt < count ; cnt++ ) {
235                 work = buffer[cnt].cv[0];
236                 buffer[cnt].cv[0] = buffer[cnt].cv[1];
237                 buffer[cnt].cv[1] = work;
238                 work = buffer[cnt].cv[2];
239                 buffer[cnt].cv[2] = buffer[cnt].cv[3];
240                 buffer[cnt].cv[3] = work;
```

```
LINE #  SOURCE TEXT
241                 buffer[cnt].fval = buffer[cnt].fval / 4;
242
243
244         }
245 }
246
247
248 /*****************************************************/
249 /*                                                   */
250 /*         This function convert                     */
251 /*    DEC G FLOAT to IEEE double precision float     */
252 /*                                                   */
253 /*****************************************************/
254 double_g_dec_to_ieee(buffer,count)
255 union all_myghty {
256         double  fval;
257         int     ival[2];
258         char    cv[8];
259 }
260 *buffer;
261
262 int       count;
263 {
264         char      work;
265         int       cnt;
266
267         /********************** swap byte ***********************/
268         for( cnt = 0 ; cnt < count ; cnt++ ) {
269                 work = buffer[cnt].cv[0];
270                 buffer[cnt].cv[0] = buffer[cnt].cv[1];
271                 buffer[cnt].cv[1] = work;
272                 work = buffer[cnt].cv[2];
273                 buffer[cnt].cv[2] = buffer[cnt].cv[3];
274                 buffer[cnt].cv[3] = work;
275                 work = buffer[cnt].cv[4];
276                 buffer[cnt].cv[4] = buffer[cnt].cv[5];
277                 buffer[cnt].cv[5] = work;
278                 work = buffer[cnt].cv[6];
279                 buffer[cnt].cv[6] = buffer[cnt].cv[7];
280                 buffer[cnt].cv[7] = work;
281                 buffer[cnt].fval = buffer[cnt].fval / 4;
282         }
283 }
284 /*****************************************************/
285 /*                                                   */
286 /*         This function convert                     */
287 /*    DEC long int to IEEE long int float            */
288 /*                                                   */
289 /*****************************************************/
290
291
292 long_dec_to_ieee(buffer,count)
293 union all_myghty {
294         long    ival;
295         char     cv[4];
296 }
297 *buffer;
298
299 int               count;
300 {
301         char      work;
302         int       cnt;
303         /********************** swap byte ***********************/
304         for( cnt = 0 ; cnt < count ; cnt++ ) {
305                 work = buffer[cnt].cv[0];
306                 buffer[cnt].cv[0] = buffer[cnt].cv[3];
307                 buffer[cnt].cv[3] = work;
308                 work = buffer[cnt].cv[1];
309                 buffer[cnt].cv[1] = buffer[cnt].cv[2];
310                 buffer[cnt].cv[2] = work;
311         }
312 }
313
314 /*****************************************************/
315 /*                                                   */
316 /*         This function convert                     */
317 /*         DEC SHORT int to IEEE SHORT int           */
318 /*                                                   */
319 /*****************************************************/
320
321
322 short_dec_to_ieee(buffer,count)
323 union all_myghty {
324         short    ival;
325         char     cv[2];
326 }
327 *buffer;
328
329 int               count;
330 {
331         char      work;
332         int       cnt;
333
334         /********************** swap byte ***********************/
335         for( cnt = 0 ; cnt < count ; cnt++ ) {
336                 work = buffer[cnt].cv[0];
337                 buffer[cnt].cv[0] = buffer[cnt].cv[1];
338                 buffer[cnt].cv[1] = work;
339         }
340 }
341
342
343
344
345
346
347
348
349
350
```

LINE # SOURCE TEXT

```
  1 #include "config.h"
  2 #include "imhmm.h"
  3 #include "log.h"
  4
  5
  6 adapt_hmm( hmm, book, floor, top, hist, code )
  7 IMHMM   *hmm;
  8 int     book, top, *code;
  9 double  floor;
 10 float   *hist;
 11 {
 12   register  i, j, k;
 13   int       ptr, vec;
 14   float     px, buf[MAX_ALPHABET], sum;
 15
 16   for( i = 0; i < hmm->num_omatrix; i++ )
 17    {
 18      ptr = hmm->num_voctor[book] * i;
 19
 20      sum = 0.0;
 21      for( j = 0; j < hmm->num_vector[book]; j++ )
 22       {
 23         vec = code[top*j];
 24         px = hmm->output_prob[book][ptr+vec] * hist[top*j];
 25
 26         for( k = 1; k < top; k++ )
 27          {
 28            vec = code[top*j+k];
 29            px += hmm->output_prob[book][ptr+vec] * hist[top*j+k];
 30          }
 31
 32         if( px > (float)floor ) buf[j] = px;
 33         else                    buf[j] = floor;
 34
 35         sum += buf[j];
 36       }
 37 #if 0
 38      printf( "sum= %e\n", sum ); fflush(stdout);
 39 #endif
 40      for( j = 0; j < hmm->num_vector[book]; j++ )
 41         hmm->output_prob[book][ptr+j] = buf[j] / sum;
 42    }
 43 }
 44
 45
 46 get_hist( fn, top, size, hist, code )
 47 char   *fn;
 48 int    top, size;
 49 int    **code;
 50 float  **hist;
 51 {
 52   float  *hist0;
 53
 54   read_hist( fn, size, &hist0 );
 55
 56 #if 0
 57   for( i = 0; i < 5; i++ )
 58    for( j = 0; j < size; j++ )
 59      printf( "hist[%3d][%3d]= %f\n", i, j, *(hist0+i+j*size) );
 60 #endif
 61
 62   sort_hist( hist0, size, top, hist, code );
 63   free( hist0 );
 64 }
 65
 66
 67 read_hist( fn, size, hist )
 68 char   *fn;
 69 int    size;
 70 float  **hist;
 71 {
 72   int     rbl, rb2, fd;
 73   float   *buf;
 74   char    *malloc( );
 75
 76   fd = open( fn, 0 );
 77   if( fd < 0 ){ printf( "can't open %s\n", fn ); exit(1); }
 78
 79   rbl = size * size * sizeof(float);
 80
 81   buf = (float*)malloc( rbl );
 82   if( buf == NULL ) { printf( "can't allocate histbuf\n" ); exit(1); }
 83
 84   rb2 = read( fd, buf, rbl );
 85   if( rbl != rb2 )
 86    {
 87      printf( "read_hist: error rbl(%d) != rb2(%d)%s\n", rbl, rb2, fn );
 88      exit(1);
 89    }
 90
 91   close( fd );
 92
 93 #if DEC_TO_IEEE
 94   float_dec_to_ieee( buf, rbl/sizeof(float) );
 95 #endif
 96
 97   *hist = buf;
 98 }
 99
100
101 #define DEBUG   0
102
103 sort_hist( hist0, size, top, hist, code )
104 float *hist0, **hist;
105 int   size, top, **code;
106 {
107   int    i, j, ptr1, ptr2;
108   float  *histbuf, sum;
109   int    *codebuf;
110   char   *malloc( );
111
112   histbuf = (float*)malloc( size * top * sizeof(float) );
113   if( histbuf == NULL ) { printf( "can't allocate histbuf\n" ); exit(1); }
114
115   codebuf = (int*)malloc( size * top * sizeof(int) );
116   if( codebuf == NULL ) { printf( "can't allocate codebuf\n" ); exit(1); }
117
118   /****** sorting *******/
119
120   for( i = 0; i < size; i++ )
```

LINE # | SOURCE TEXT

```
121     {
122       ptr1 = i * top;
123       for( j = 0; j < top; j++ ) histbuf[ptr1+j] = -10000.0;
124
125       for( j = 0; j < size; j++  )
126        {
127          ptr2 = j * size + i;
128          sort_large( top, histbuf+ptr1, codebuf+ptr1, hist0[ptr2], j );
129        }
130
131       /******* normalization *******/
132
133       sum = 0.0;
134       for( j = 0; j < top; j++  )
135        {
136          if( histbuf[ptr1+j] < 0.0 )
137           {
138             printf( "hist[%3d][%3d]= %f < 0.0\n",
139                      i, codebuf[ptr1+j], histbuf[ptr1+j] );
140             exit(1);
141           }
142          sum += histbuf[ptr1+j];
143        }
144
145       if( sum != 0.0 )
146        {
147          for( j = 0; j < top; j++  )
148           {
149             histbuf[ptr1+j] /= sum;
150 #if DEBUG
151          printf( "hist[%3d][%3d]= %f\n", i, codebuf[ptr1+j], histbuf[ptr1+j] );
152 #endif
153           }
154        }
155 #if DEBUG
156       printf( "\n" );
157 #endif
158
159     }
160
161   *hist = histbuf;
162   *code = codebuf;
163 }
164
165
166 sort_large( top_num, val, cod, val_in, cod_in )
167 int    top_num, cod[ ], cod_in;
168 float  val[ ], val_in;
169 {
170   int  i, k;
171
172   k = top_num + 1;
173   for( i = 0; i < top_num; i++ )
174     if( val_in > val[i] ) { k = i; break; }
175
176   if( k < top_num )
177    {
178      for( i = top_num-1; i > k; i-- )
179       {
180         val[i] = val[i-1];
181         cod[i] = cod[i-1];
182       }
183      val[k] = val_in;
184      cod[k] = cod_in;
185    }
186 }
```